Universidad Carlos III de Madrid

Grado en Ingeniería de Sistemas de Comunicaciones

2015-2016

Trabajo Fin de Grado

DISEÑO E IMPLEMENTACIÓN DE UNA INFRAESTRUCTURA BIG DATA PARA ANÁLISIS DE MERCADOS FINANCIEROS

Autor: Miguel Aller Camino

Tutor: Pablo Basanta Val

Leganés, julio de 2016

Título: Diseño e implementación de una infraestructura Big Data para análisis de mercados financieros

Autor: Miguel Aller Camino

Director: Pablo Basanta Val

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 5 de Julio de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de ___.

SECRETARIO VOCAL PRESIDENTE

Agradecimientos

En primer lugar, quiero agradecer a mi madre, a mi padre y a mi hermana el apoyo incondicional que me han dado a lo largo de toda mi vida. Gracias por haberme dado la oportunidad de poder desarrollarme a nivel educativo de la manera que lo habéis hecho, con tanto esfuerzo y sacrificio. Gracias también por haberme inculcado unos valores basados en la solidaridad, generosidad, honradez y esfuerzo, por vosotros sois lo que he llegado a ser hoy, tanto en el aspecto profesional como en el personal. Gracias, gracias y gracias por perseguir cada día desde hace seis años el mismo sueño que yo, por fin es una realidad. Os quiero mucho.

Gracias a mi abuela y a mi tío, por empujar siempre desde 'nuestra aldea', cada aliento, cada palabra, por estar ahí siempre que os he necesitado, gracias de corazón. Sin vosotros hubiera sido imposible.

Gracias a Michel, por ser uno más de la familia desde el primer día y por el apoyo en esta última etapa, cada mensaje era un punto más de convicción para lograr el objetivo. ΥΓΕΙΑ

También agradecer a todos mis amigos que han estado presentes a lo largo de este tiempo, tanto de Vigo como de Madrid, por haber hecho de éste un camino más sencillo. En especial a Alex, tu presencia en Madrid ha significado mucho desde que nos conocimos, gracias. Por supuesto, agradecer a mis compañeros de universidad, por tantas horas dedicadas juntos para alcanzar un objetivo, algunos ya sois amigos y estaréis ahí siempre.

Gracias a Pablo, mi tutor, por todas esas horas resolviéndome dudas y ayudando en todo lo que le pedía. Gracias por darme la oportunidad de hacer este proyecto juntos, ha sido una experiencia muy positiva, y tras mucha dedicación y esfuerzo, hemos logrado que haya sido de éxito.

Gracias a mi perro, Nuno, y al recién llegado Simba, sin esos momentos de desconexión y risas la vida no sería lo mismo. Gracias Lucky, por tantos lametones y tanto cariño, nunca nos olvidaremos de ti allí donde estés.

Por último, un gracias muy especial, a mi novia Laura. Gracias por aguantarme, por apoyarme en cada momento, por sacarme una sonrisa cuando parecía imposible, por ayudarme siempre que te lo he pedido, por tantos momentos juntos. Gracias por conseguir lo que parecía imposible y hacerlo desde tan lejos. Gracias por aparecer en mi vida y empujarme a alcanzar el objetivo con decisión, esto también es tuyo, muy tuyo. Te quiero mucho.

Agradecer a todas esas persona que aunque no haya citado han contribuido y ayudado, deseándome lo mejor.

Resumen

Smartphones, wearables, sensores y todo tipo de empresas y organizaciones generan millones de datos al día. De la necesidad de la organización y el procesado de esos datos surge el término Big Data, a través del cual podemos realizar dichas tareas de una manera rápida y eficiente, que con las herramientas de procesado existentes hasta el momento no hubiera sido posible.

El mundo del Big Data ha crecido enormemente en los últimos años, tanto en volumen de negocio como en herramientas disponibles para su desarrollo, permitiendo grandes avances y creando un sector completamente nuevo. Son muchas las empresas que recurren a estas nuevas tecnologías para obtener resultados de grandes conjuntos de datos, ya que el propio dato es el que contiene un valor oculto, que solo aparece al haber sido procesado, dotándolo de sentido.

La importancia de la economía en la vida de las personas es muy grande, a través de los mercados financieros se manejan los flujos de capitales de los inversores que apuestan por las grandes empresas, modificando el funcionamiento de los mercados bursátiles y por tanto de la economía. De la necesidad del análisis de los datos procedentes de esos mercados y aprovechando las ventajas del Big Data, en este trabajo de fin de grado se han querido conjugar ambos campos con el objetivo de obtener herramientas de inversión predictivas en un sistema eficaz, eficiente, rápido y escalable, adaptándose a diferentes escenarios.

Para la consecución de los objetivos fijados se han empleado dos tecnologías utilizadas en el mundo Big Data, Hadoop y Hive, las cuáles, y como se verá a lo largo de este proyecto, nos proporcionan las herramientas necesarias para el tratado y procesado de datos.

Una vez finalizado el proyecto, se podrá recabar información de diferentes mercados financieros, introducirlos en el sistema Big Data diseñado para su procesamiento y análisis a partir de algoritmos de inversión, los cuáles nos aportaran una serie de resultados. Además, contaremos con diferentes alternativas de diseño, todas ellas orientadas a satisfacer necesidades provenientes de diferentes posibilidades de uso.

Abstract

Smartphones, wearables, sensors and all type of companies and organizations generate countless amounts of data per day. From the need for these companies to process such a massive amount of data, the term and concept of Big Data has been created. The new Big Data systems allow a faster and more efficient data processing in comparison to the traditional processing tools that existed and were available in the market until now.

Big Data has grown tremendously in the last few years, both in terms of volume of operations and business revenue as well as in the number of tools available for its development. This has allowed significant progress and the creation of a completely new sector. There are many companies that turn to these new technologies in order to be able to process big amounts of data and obtain interesting and reliable results and outcomes from the analysis of these data. We also need to consider that hidden values exist within the data and only surface when processing the data on an appropriate manner and by using the adequate tools that will provide a sense to them and make possible the extraction of trustworthy results for further analysis.

It is undeniable that the importance of the economy on people's life is huge. Private capitals from the investors flow into the financial markets and different stock exchanges altering stock prices significantly depending on the mood and appetite of the investors due to other external factors such as economic news, rumours, decisions made by governmental institutions or central banks on the different regions, or simply the expectations on the market's future reactions/movements, among others. All these also have then an impact on the real economy. From the need of analysing the data coming from these markets and by making the most out of the Big Data technology and its benefits, we will focus our end-of-degree study on the testing and implementation of new tools and technologies for analysis of Big Data (massive amount of data) and investment forecasting, with special focus on the financial services industry, that would help to extract accurate and reliable results for further analysis of the financial markets on an efficient, effective, fast, versatile and scalable way.

In order to reach this goal, we will use during our project two main technologies used on the world of Big Data: Hadoop and Hive, who provide the required tools needed for the treatment and processing of big volumes of data.

Once the project has been finalized, we will be able to extract information from the different financial markets and stock exchanges and load them into the Big Data system that has been design in order to process and analyse from the results extracted from the application of algorithmic trading. Furthermore, we will consider different scenarios and build alternative designs in order to adjust our infrastructure to the different objectives

and requirements of the user, also in other fields aside of the financial markets, and suffice their needs.

Tabla de contenidos

Agradecimientos 3
Resumen 4
Abstract 5
Índice de Figuras 11
Índice de Tablas 14
BLOQUE I 15
Introducción 15
1. Introducción y objetivos 16
1.1. Contexto y motivación 16
1.2. Objetivos 17
1.3. Estructura del documento 18
BLOQUE II 20
Planteamiento del problema 20
2. 1. Estado del arte 21
2.1.1. Big Data 21
2.1.1.1. Introducción 21
2.1.1.2. Visión y necesidad 21
2.1.1.3. Situación 22
2.1.1.4. Veracidad de los datos 23
2.1.1.5. Minería de datos 23
2.1.2. Computación distribuida 24
2.1.2.1. Introducción 24
2.1.2.2. Tipos 25
2.1.3. Infraestructura 27
2.1.3.1. Apache Hadoop 27
2.1.3.2. Apache Hive 31

2.1.4. Mercados Financieros 31
2.1.4.1. Introducción 31
2.1.4.2. Mercados financieros destacados 31
2.1.4.3. La importancia del análisis de datos ágil y eficaz en el mundo económico 32
2.2. Marco regulador 32
2.2.1. Marco regulador técnico 32
2.2.2. Privacidad de datos 33
BLOQUE III 36
Trabajo realizado 36
3. Diseño del sistema 37
3.1. Introducción 37
3.2. Bloques Funcionales 37
3.3. Arquitectura del sistema 39
3.4.1. Sistema Pseudo-Distribuido 39
3.4.2. Sistema Distribuido 40
4. Implementación 46
4.1. Introducción 46
4.2. Preparación del entorno de trabajo 46
4.2.1. Instalación Linux 47
4.2.2. Instalación Hadoop 49
4.2.3. Instalación Hive 60
4.3. Obtención de los datos 62
4.3.1. Introducción 62
4.3.2. Fuentes de información 62
4.3.3. Procedimiento 63
4.4. Transformación y almacenamiento de los datos 64
4.4.1. Introducción 64

4.4.2. Procedimiento 65
4.4. Análisis y tratado de datos mediante scripts HiveQL 66
4.5.1. Algoritmo básico de estrategia de inversión 66
4.5.2. Algoritmo avanzado de estrategia de inversión 75
5. Uso del sistema 79
5.1. Arranque de las máquinas 79
5.2. Arranque de Hadoop 79
5.2.1. Verificación del correcto funcionamiento de los demonios 81
5.2.2. Gestión del sistema Hadoop a través de interfaz web 81
5.3. Arranque de Hive 83
5.4. Utilización de scripts HQL 83
5.5. Apagado del sistema 84
6. Pruebas y resultados 85
6.1. Introducción 85
6.2. Entorno de prueba 85
6.2.1. Configuración pseudo-distribuida 85
6.2.2. Cluster doméstico 86
6.2.3. Cluster Universitario 88
6.3. Efectividad de los algoritmos generados 89
6.3.1. Efectividad del algoritmo básico 89
6.3.2. Efectividad del algoritmo avanzado 90
6.4. Eficiencia de la infraestructura 90
6.4.1. Introducción 90
6.4.2. Datos sintéticos 91
6.4.3. Modo pseudo-distribuido 93
6.4.4. Cluster doméstico 96
6.4.5. Cluster universitario 102

6.4.6. Comparativa eficiencia arquitectura Hadoop 108
BLOQUE I V 113
Conclusiones y líneas futuras de trabajo 113
7. Conclusiones y líneas futuras de trabajo 114
7.1. Conclusiones 114
7.2. Futuras líneas de trabajo 114
BLOQUE V 115
Planificación y presupuesto 115
8. Planificación y presupuesto 116
8.1. Planificación 116
8.1.1. Fases de desarrollo 116
8.1.2. Diagrama de Gantt 118
8.2. Presupuesto 119
8.2.1. Medios empleados 119
8.2.2. Presupuesto 120
Bibliografía 123
Anexo I: Summary 125
Anexo II: Procedimiento de configuración cluster universitario 157

Índice de Figuras

Figura 1: Esquema bloques fundamentales sistema Big Data 17
Figura 2: Previsión de crecimiento mundo Big Data 22
Figura 3: Tipos de datos más comunes tratados por el Big Data 23
Figura 4: Metodología presente en un proceso de minería de datos 24
Figura 5: Esquema computación distribuida 25
Figura 6: Visión general de una arquitectura cluster incluyendo todos sus componentes 26
Figura 7: Módulos principales Apache Hadoop 27
Figura 8: Procesos (daemons) ejecutados en Apache Hadoop 28
Figura 9: Visión general arquitectura HDFS 29
Figura 10: Proceso de escritura en HDFS 29
Figura 11: Proceso de lectura en HDFS 30
Figura 12: Esquema general del funcionamiento de Mapreduce 30
Figura 13: Esquema general del sistema Big Data incluyendo los bloques funcionales 38
Figura 14: Esquema de los tipos de sistema Big Data planteados 39
Figura 15: Visión general del sistema pseudo-distribuido incluyendo los procesos que en él participan 40
Figura 16: Arquitectura cluster doméstico diseñada 41
Figura 17: Configuración aplicada al cluster domestico incluyendo los procesos Hadoop 42
Figura 18: Arquitectura cluster universitario diseñada 43
Figura 19: Configuración aplicada al cluster universitario incluyendo los procesos Hadoop 44
Figura 20: Creación de un usb bootable a través del programa Rufus 48
Figura 21: Comandos instalación Java en Linux 49
Figura 22: Comandos instalación Rsync en Linux 49
Figura 23: Comandos creación usuario y grupo en Linux 50
Figura 24: Comandos instalación Open-SSH en Linux 50
Figura 25: Comandos configuración SSH en Linux 50
Figura 26: Comandos instalación Hadoop en Linux 51
Figura 27: Creación carpetas temporales HDFS configuración pseudo-distribuida 51
Figura 28: Fichero de configuración del usuario hduser .bashrc 52
Figura 29: Fichero de configuración core-site.xml configuración pseudo-distribuida 52
Figura 30: Fichero de configuración hdfs-site.xml configuración pseudo-distribuida 53
Figura 31: Fichero de configuración hadoop-env.sh configuración pseudo-distribuida 53
Figura 32: Fichero de configuración yarn-site.xml configuración pseudo-distribuida 53
Figura 33: Fichero de configuración mapred-site.xml configuración pseudo-distribuida 54
Figura 34: Fichero de configuración hosts 55
Figura 35: Configuración IP local en Linux para cluster doméstico 55
Figura 36: Configuración permisos de usuario en Linux 56
Figura 37: Edición nombre de localhost 56
Figura 38: Fichero de configuración core-site.xml configuración multinodo 57
Figura 39: Fichero de configuración hdfs-site.xml configuración multinodo 57
Figura 40: Fichero de configuración yarn-site.xml configuración multinodo 58
Figura 41: Fichero de configuración mapred-site.xml configuración multinodo 58

Figura 42: Fichero de configuración maestro configuración multinodo 58
Figura 43: Fichero de configuración esclavos configuración multinodo 58
Figura 44: Copia de archivos Hadoop desde el maestro hacia los esclavos 59
Figura 45: Creación carpetas para archivos temporales HDFS multinodo en maestro 59
Figura 46: Creación carpetas para archivos temporales HDFS multinodo en los esclavos 59
Figura 47: Distribución claves ssh entre máquinas 60
Figura 48: Instalación Hive 61
Figura 49: Configuración de las variables de entorno para Hive 61
Figura 50: Creación de carpetas Hive en HDFS 61
Figura 51: Ejemplo de datos procedente de un archivos csv de mercados financieros 64
Figura 52: Estructura de tablas en Hive para almacenamiento de datos 65
Figura 53: Estructura de bases de datos en Hive para el almacenamiento de tablas 65
Figura 54: Creación de bases de datos en Hive 65
Figura 55: Creación de tablas e importación de datos en Hive 66
Figura 56: Estrategia de inversión algoritmo básico 67
Figura 57: Algoritmo de inversión básico Nasdaq 70
Figura 58: Algoritmo de inversión básico Ftse 71
Figura 59: Algoritmo de inversión básico Nikkei 72
Figura 60: Algoritmo de inversión básico Ibex 73
Figura 61: Algoritmo de inversión básico simplificado 74
Figura 62: Estrategia de inversión avanzada 75
Figura 63: Algoritmo estrategia de inversión avanzada 77
Figura 64: Secuencia de inicio procesos Hadoop desfasado 80
Figura 65: Secuencia de inicio procesos HDFS 80
Figura 66: Secuencia de inicio procesos YARN 80
Figura 67: Supervisión sistema Hadoop a través de interfaz web 82
Figura 68: Supervisión nodos del cluster a través de interfaz web 82
Figura 69: Secuencia de inicio Hive 83
Figura 70: Efectividad algoritmos de inversión básico 89
Figura 71: Adaptación del algoritmo básico simplificado para datos sintéticos 93
Figura 72: Eficiencia configuración pseudo-distribuida 95
Figura 73: Nodos activos en el cluster doméstico durante la ejecución de Hadoop 96
Figura 74: Eficiencia configuración multinodo (1 nodo) - cluster doméstico 99
Figura 75: Eficiencia configuración multinodo (2 nodos) – cluster doméstico 100
Figura 76: Eficiencia configuración multinodo (3 nodos)- cluster doméstico 101
Figura 77: Nodos activos en cluster universitario durante la ejecución de Hadoop 102
Figura 78: Eficiencia configuración multinodo (2 nodos) – cluster universitario 103
Figura 79: Eficiencia configuración multinodo (4 nodos) – cluster universitario 104
Figura 80: Eficiencia configuración multinodo (8 nodos) – cluster universitario 105
Figura 81: Eficiencia configuración multinodo (16 nodos) – cluster universitario 106
Figura 82: Eficiencia configuración multinodo (24 nodos) – cluster universitario 107
Figura 83: Comparativa tiempos de procesamiento configuración pseudo-distribuida vs cluster doméstico 108
Figura 84: Comparativa registros/segundo configuración pseudo-distribuida vs cluster doméstico 109

Figura 85: Comparativa tiempos de procesamiento cluster universitario 110
Figura 86: Comparativa registros/segundo cluster universitario 111
Figura 87: Comparativa registros/segundo entre clusteres 112
Figura 88: Comparativa tiempos de procesamiento entre clusteres 112
Figura 89: Diagrama de Gantt - fases de desarrollo del proyecto 118

Índice de Tablas

Tabla 1: Estándares y organizaciones regulación Big Data 33
Tabla 2: Fuentes y detalles de la información empleada 63
Tabla 3: Estructura archivo csv mercados financieros Yahoo Finance 64
Tabla 4: Filtrado datos originales 68
Tabla 5: Relación mercados financieros - scripts creados 68
Tabla 6: Ejecución de procesos en configuración pseudo-distribuida 81
Tabla 7: Ejecución de procesos en configuración multinodo 81
Tabla 8: Listado completo de scripts empleados 83
Tabla 9: Sentencias para el apagado del sistema 84
Tabla 10: Especificaciones maestro en sistema pseudo-distribuido 85
Tabla 11: Especificaciones esclavo 1 en cluster doméstico 86
Tabla 12: Especificaciones esclavo 2 en cluster doméstico 87
Tabla 13: Especificaciones esclavo 3 en cluster doméstico 87
Tabla 14: Especificaciones equipos en cluster universitario 88
Tabla 15: Efectividad algoritmo básico de inversión para mercados financieros 89
Tabla 16: Efectividad algoritmo de inversión avanzado 90
Tabla 17: Listado de archivos generados a partir de datos sintéticos 91
Tabla 18: Tiempo de procesamiento algoritmo básico sistema pseudo-distribuido 93
Tabla 19: Tiempo de procesamiento algoritmo avanzado sistema pseudo-distribuido 94
Tabla 20: Tiempos de procesamiento datos sintéticos sistema pseudo-distribuido 95
Tabla 21: Tiempo de procesamiento algoritmo básico cluster doméstico (1 nodo) 97
Tabla 22: Tiempo de procesamiento algoritmo avanzado cluster doméstico (1 nodo) 97
Tabla 23: Tiempo de procesamiento algoritmo básico cluster doméstico (2 nodos) 97
Tabla 24: Tiempo de procesamiento algoritmo avanzado cluster doméstico (2 nodos) 98
Tabla 25: Tiempo de procesamiento algoritmo básico cluster doméstico (3 nodos) 98
Tabla 26: Tiempo de procesamiento algoritmo avanzado cluster doméstico (3 nodos) 98
Tabla 27: Tiempos de procesamiento datos sintéticos cluster doméstico (1 nodo) 99
Tabla 28: Tiempos de procesamiento datos sintéticos cluster doméstico (2 nodos) 100
Tabla 29: Tiempos de procesamiento datos sintéticos cluster doméstico (3 nodos) 101
Tabla 30: Tiempos de procesamiento datos sintéticos cluster universitario (1 nodo) 103
Tabla 31: Tiempos de procesamiento datos sintéticos cluster universitario (4 nodos) 104
Tabla 32: Tiempos de procesamiento datos sintéticos cluster universitario (8 nodos) 105
Tabla 33: Tiempos de procesamiento datos sintéticos cluster universitario (16 nodos) 106
Tabla 34: Tiempos de procesamiento datos sintéticos cluster universitario (24 nodos) 107
Tabla 35: Detalle de horas empleadas por fases de desarrollo 120
Tabla 36: Costes de los medios personales 121
Tabla 37: Costes de los medios materiales 121
Tabla 38: Costes de los medios inmateriales 122
Tabla 39: Costes de otros medios 122
Tabla 40: Coste total del proyecto 122

BLOQUE I

Introducción

1.Introducción y objetivos

1.1. Contexto y motivación

En los últimos años, el tráfico y generación de datos ha aumentado de una forma muy notoria por el incremento del número de dispositivos y de personas con posibilidades de acceso a la red. La era del Smartphone, las tablets, más recientemente los wearables y el internet de las cosas hacen que la visión general del análisis de datos existente quedase desfasada por la imposibilidad de asumir la enorme demanda que estos nuevos datos suponen.

Las empresas y diferentes organizaciones buscan a partir de esta gran cantidad de datos, obtener una ventaja competitiva frente a otras empresas u organizaciones, aprovechando la utilidad de esa información recibida, buscan trazar estrategias empresariales para aumentar sus beneficios o en caso de organismos públicos su buen funcionamiento.

Dentro del mundo de los mercados financieros, la importancia de los datos es muy grande, los fondos de inversión, banca privada, bancos tradicionales e incluso los gobiernos necesitan herramientas de análisis de datos eficaces, potentes y rápidas que les proporcionen unos resultados veraces e inmediatos, que hoy por hoy, con las herramientas tradicionales comienza a resultar más complicado por el coste computacional que ello requiere.

A partir de las necesidades anteriores, este proyecto se enmarca dentro del sector financiero y de los mercados de valores globales, analizando tendencias históricas para generar algoritmos de inversión rentables a partir de los cuáles obtener rendimientos de capitales interesantes para los interesados antes citados.

A partir de datos libres accesibles en cualquier momento [1], interactuamos con un sistema Big Data basado dos tecnologías; Apache Hadoop [2] y Apache Hive [3] . Una vez obtenidos los datos, se cargan en el sistema, se procede al análisis de esos datos y se obtienen los resultados. El procesamiento de esos datos se realiza mediante los algoritmos de inversión antes citados que se apoyan en comandos Hive.

El sistema Big Data cuenta con diferentes soluciones posibles que se expondrán a lo largo de este documento, situándonos en diferentes escenarios que harán que la infraestructura nos aporte una eficiencia u otra.

Por tanto, la visión de conjunto del proyecto a desarrollar es la expuesta a continuación, dónde se muestran los bloques fundamentales del sistema Big Data con su funcionalidad.

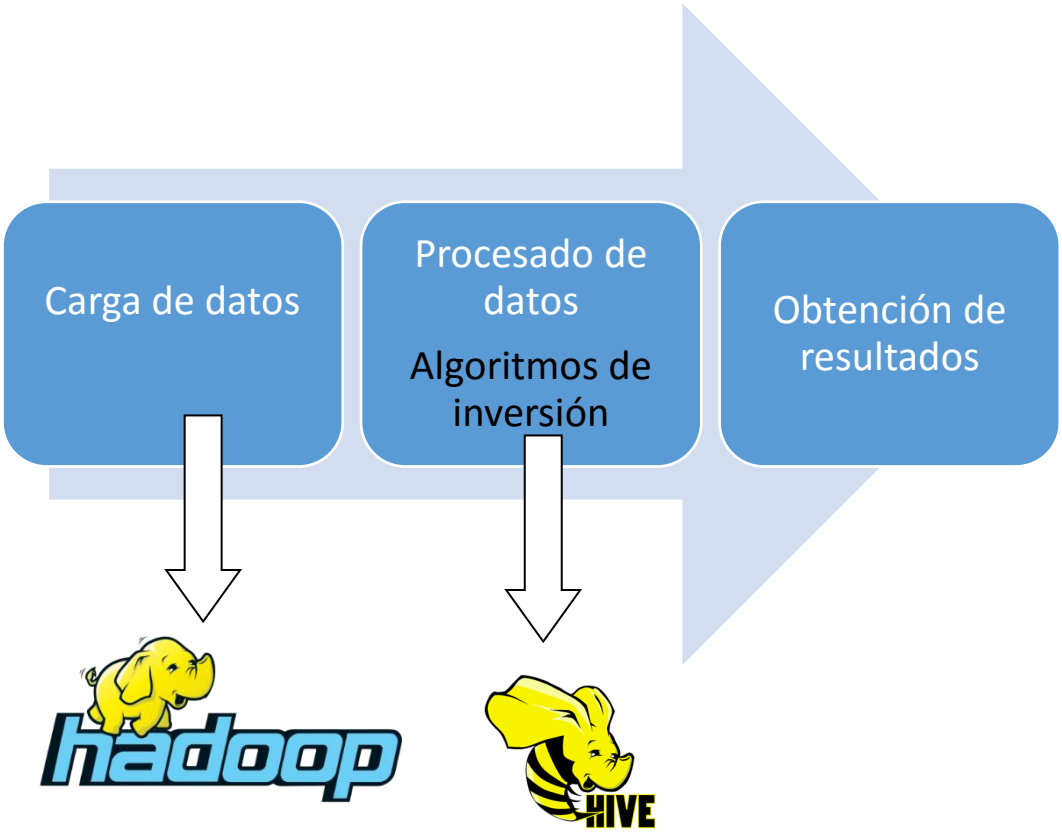


Figura 1: Esquema bloques fundamentales sistema Big Data

1.2. Objetivos

El objetivo principal es el diseño e implementación de un sistema Big Data eficiente, escalable y polivalente basado en Hadoop para el análisis de mercados financieros apoyándose en Hive.

Para alcanzar dicho objetivo el proyecto requiere de los siguientes puntos:

- Análisis y estudio de las diferentes tecnologías necesarias para implementar el sistema (Apache Hadoop, Apache Hive).
- Estudio de las necesidades de los mercados financieros.
- Diseño del sistema Big Data con diferentes configuraciones.
- Implementación en diferentes entornos del sistema Big Data diseñado.
- Obtención de resultados de los diferentes algoritmos de inversión financiera generados.

- Aprendizaje del uso del sistema implementado así como el uso de scripts.
- Análisis de la eficiencia del sistema diseñado bajo las diferentes configuraciones y entornos.
- Obtener conclusiones y futuras líneas de trabajo del sistema implementado a partir de los datos obtenidos.
- Realizar un presupuesto detallado que nos permita conocer la magnitud económica del proyecto.

1.3. Estructura del documento

A continuación se presenta la estructura del presente documento para ofrecer una visión general del mismo con el objetivo de facilitar su seguimiento y lectura.

1. BLOQUE I: Introducción

   En este bloque se expone el contexto y motivación del proyecto realizado, los objetivos del mismo y la estructura del documento del proyecto.

2. BLOQUE II: Planteamiento del problema
   - Estado del arte: Exposición de las tecnologías empleadas en la implementación del sistema Big Data así como del entorno de los mercados financieros.
   - Marco Regulador: Detalle de las leyes aplicables al sistema implementado y a la privacidad de datos.

3. BLOQUE III: Trabajo realizado
   - Diseño del sistema: Exposición del sistema implementado, incluyendo diferentes configuraciones y escenarios.
   - Implementación: Se incluye la instalación de todos los componentes necesarios para el funcionamiento del sistema diseñado. Incluye también el procedimiento de obtención, transformación y análisis de datos.
   - Uso del sistema: Se muestra detalladamente el uso completo del sistema, así como de los scripts generados.
   - Pruebas y resultados: Se exponen los resultados obtenidos a partir de las diferentes pruebas realizadas, haciendo una comparativa en cuánto a eficiencia de los diferentes escenarios y configuraciones planteados.

4. BLOQUE IV: Conclusiones y líneas futuras de trabajo

   Se plantean las posibles líneas futuras de trabajo a seguir, así como las conclusiones extraídas tras la realización del proyecto.

5. BLOQUE V: Planificación y presupuesto
   - Planificación: Se detallan las fases del proyecto realizadas para alcanzar su consecución, incluyendo diagrama de fases.
   - Presupuesto: Se incluye un presupuesto detallado del proyecto.

6. BLOQUE VI: Anexos

   Incluye todos los anexos del proyecto.

BLOQUE II

Planteamiento del problema

2.1. Estado del arte

2.1.1. Big Data

2.1.1.1. Introducción

Desde las primeras formas de escritura del ser humano conocidas, la información ha sido almacenada en múltiples formatos, con el objetivo de preservarla y de dotarla de sentido.

En 1880, en Estados Unidos, debía realizarse el censo de población con los métodos que había hasta entonces, se tardó ocho años en completar la tarea. Para el censo de 1890 la estimación para realizar la misma tarea que en 1880 era de más de 10 años, es decir, no llegaría tiempo para realizar el censo de 1900, por lo que tuvieron que realizar avances en la manera de procesar tanta cantidad de datos, fue gracias a la máquina tabuladora de Hollerith, basada en tarjetas perforadas, por la cual pudieron completar la tarea. Con esto se ve la importancia del avance tecnológico en la recuperación y tratamiento de datos, clave para realizar tareas hasta entonces imposibles.

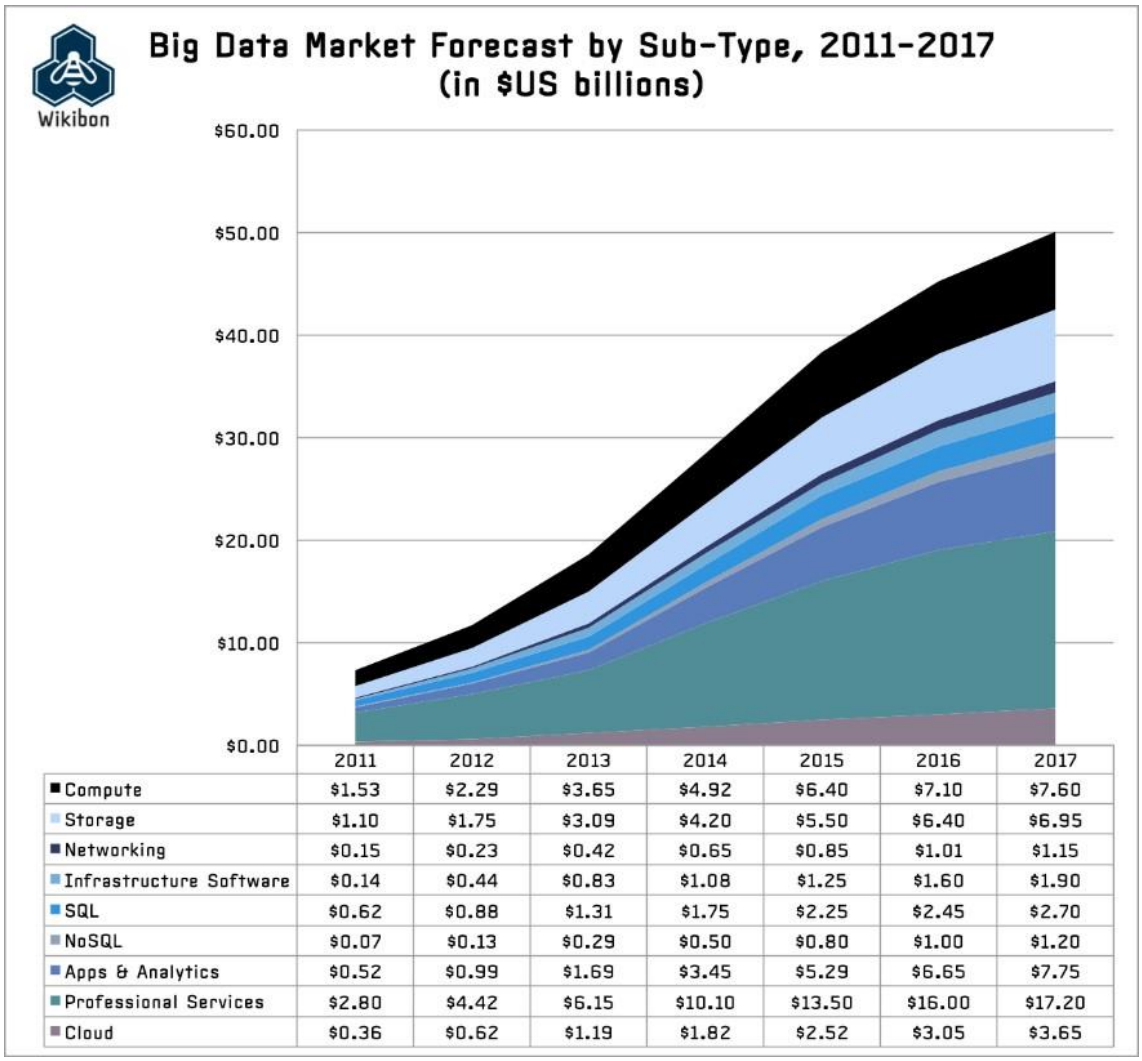
Con el auge de las tecnologías de la información y el crecimiento del sector tecnológico, se comienzan a crear y almacenar cantidades enormes de datos. La aparición de nuevos dispositivos y la mejora de las comunicaciones con millones de personas conectadas a la red, hacen que el problema se acentúe, incrementando el número de datos generados en billones. El volumen de tráfico estimado para el año 2016 equivale a 33 billones de DVDs o a 814 cuatrillones de mensajes texto [4].

De la necesidad del tratamiento y análisis de toda esa gran cantidad de información nace el término Big Data, que describe el tratamiento de aquella información, ya sea estructurada o no que no puede ser procesada utilizando herramientas tradicionales. El Big Data introduce una nueva tecnología, que permite a través de procesos y herramientas tratar, operar y usar grandes cantidades de datos.

2.1.1.2. Visión y necesidad

Las grandes compañías como Yahoo!, Oracle o Google emplean el Big data en su día a día. Google emplea esta tecnología por la necesidad de monitorizar clicks, links y nuevos contenidos en 1,5 millones de páginas por día.

Con motivo del gran auge de esta tecnología, el número de empresas interesadas en ella ha aumentado de forma significativa, aumentando por lo tanto la cifra de negocio [5] del mundo Big data, tal y como refleja la siguiente gráfica:



	2011	2012	2013	2014	2015	2016	2017
Compute	$1.53	$2.29	$3.65	$4.92	$6.40	$7.10	$7.60
Storage	$1.10	$1.75	$3.09	$4.20	$5.50	$6.40	$6.95
Networking	$0.15	$0.23	$0.42	$0.65	$0.85	$1.01	$1.15
Infrastructure Software	$0.14	$0.44	$0.83	$1.08	$1.25	$1.60	$1.90
SQL	$0.62	$0.88	$1.31	$1.75	$2.25	$2.45	$2.70
NoSQL	$0.07	$0.13	$0.29	$0.50	$0.80	$1.00	$1.20
Apps & Analytics	$0.52	$0.99	$1.69	$3.45	$5.29	$6.65	$7.75
Professional Services	$2.80	$4.42	$6.15	$10.10	$13.50	$16.00	$17.20
Cloud	$0.36	$0.62	$1.19	$1.82	$2.52	$3.05	$3.65

Figura 2: Previsión de crecimiento mundo Big Data

La necesidad, por tanto, de crear nuevas herramientas y métodos así como de profesionales que ayuden al progreso del sector es muy grande.

2.1.1.3. Situación

EL Big data viene definido principalmente por la 4V's:

-**Volumen**: Es la característica principal del Big Data, la cantidades de datos generados y procesados. No se puede establecer una medida de datos para determinar qué es Big Data y qué no lo es, ya que esta definición puede quedar obsoleta en muy poco tiempo, pero para hacernos una idea aproximada, para algunas empresas puede referirse a decenas de terabytes mientras que para otras a cientos de petabytes.

-**Velocidad:** Los datos deben ser procesados en muy poco tiempo antes de quedar desfasados por la llegada de otros nuevos, es uno de los grandes retos de cualquier infraestructura Big Data, mejorar la velocidad a partir de diferentes tipos de combinaciones.

-**Variedad:** La procedencia de los datos puede ser muy variada, desde tablets, smartphones, wearables, sensores de todo tipo, es decir, no existe una estructura definida en los datos recibidos. Los tipos de datos más habituales [6] tratados por el Big Data vienen resumidos en la siguiente figura:

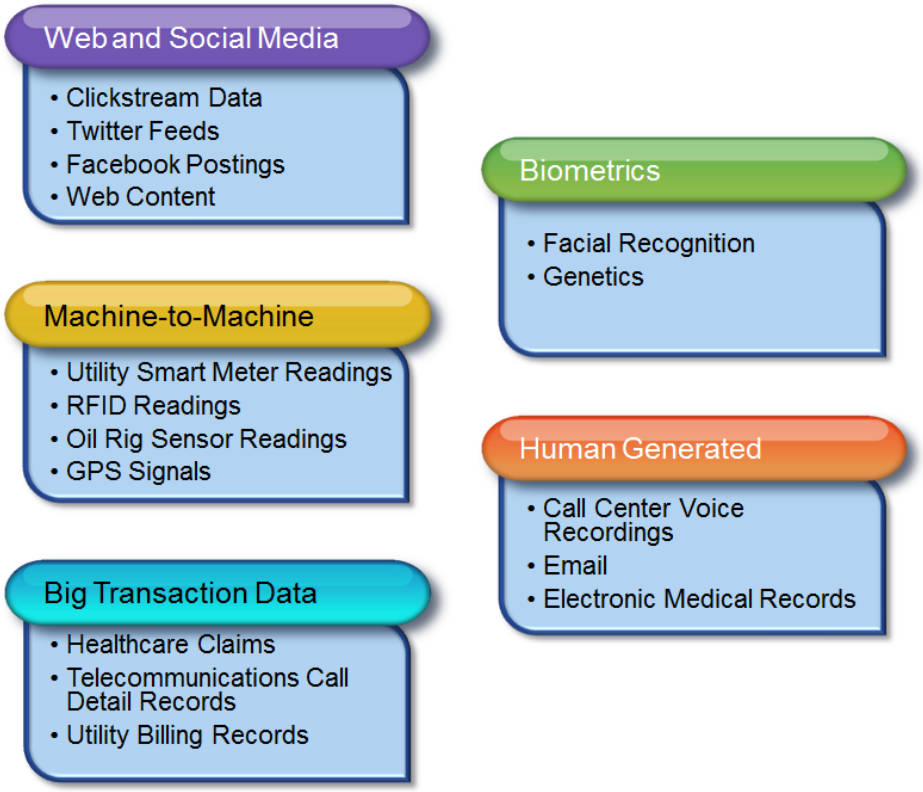


Figura 3: Tipos de datos más comunes tratados por el Big Data

-**Valor:** Los datos contienen en sí mismos valor, pero de una forma abstracta, es, una vez que han sido transformados cuando los datos cobran sentido y aportan valor al negocio.

2.1.1.4. Veracidad de los datos

La veracidad de los datos es sumamente importante en el entorno Big Data, algunas fuentes la sitúan incluso como la 5V dentro de la definición de Big Data.

Tanto la estructura de los datos recibidos como el propio dato deben ser correctos para poder partir de una buena base en el tratamiento de esos datos. A pesar de realizar un tratamiento previo al verdadero análisis el dato puede venir erróneo de la fuente, sin poder hacer nada por evitarlo.

Además, la veracidad también implica la disponibilidad futura de los datos, sobre la que en muchas ocasiones hay una gran incertidumbre.

2.1.1.5. Minería de datos

El concepto de minería de datos se refiere al proceso de detectar información procesable dentro de grandes cantidades de datos. Se pretende encontrar patrones y

tendencias existentes en los datos que con las herramientas tradicionales de análisis de datos es imposible.

Los escenarios dónde se puede aplicar la minería de datos son los siguientes:

- Pronóstico
- Agrupación
- Búsqueda de secuencias
- Recomendaciones

La metodología a seguir en un proceso de minería de datos es el siguiente [7]:

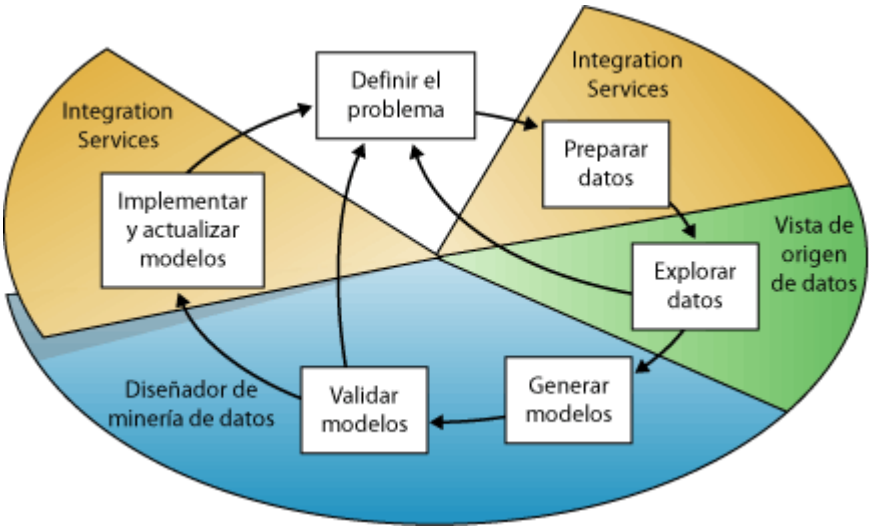


Figura 4: Metodología presente en un proceso de minería de datos

2.1.2. Computación distribuida

2.1.2.1. Introducción

Consiste en la utilización de una serie de máquinas conectadas entre sí a través de un sistema de comunicaciones con el objetivo de llevar a cabo una tarea en un tiempo menor que con una sola máquina gracias al funcionamiento conjunto de todos los recursos del sistema [8].

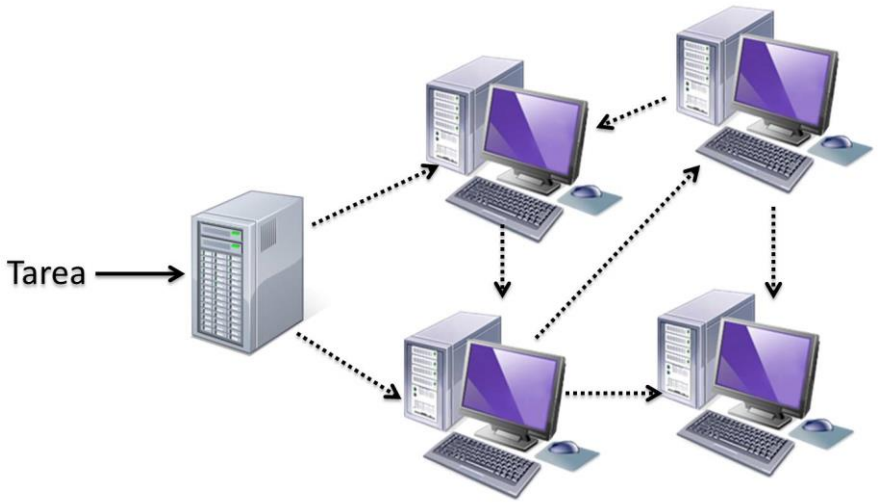


Figura 5: Esquema computación distribuida

La computación presenta una serie de ventajas e inconvenientes que pasamos a analizar a continuación:

Ventajas

- Escalabilidad: Permite la incorporación de nuevos recursos al sistema con el objetivo de cubrir nueva demanda.
- Tolerancia a fallos: Al ser sistemas con varias máquinas, el fallo de una de ellas no imposibilita continuar utilizando el sistema.
- Diferentes tipos de máquina: Posibilidad de emplear equipos diferentes dentro de la misma red.

Inconvenientes

- Protocolos: Muchos protocolos no permiten el uso de la computación distribuida limitando a través de cortafuegos.
- Seguridad: Al distribuirla tarea entre todos los equipos puede haber problemas de interceptación de mensajes en la red.

2.1.2.2. Tipos

La computación distribuida en el caso del entorno Big Data puede presentar dos tipos de configuraciones, la pseudo-distribuida y la multinodo.

2.1.2.2.1. Pseudo-Distribuida

En esta configuración tanto el equipo que actúa como administrador o maestro como el que actúa como equipo secundario o nodo esclavo se encuentran en la misma máquina.

El objetivo de este tipo de configuración, es en su mayoría, el de realizar entornos de prueba o pequeños entornos de trabajo.

2.1.2.2.2. Cluster o configuración multinodo

Se define un cluster como el conjunto de computadoras construidas mediante la utilización de componentes de hardware comunes y que se comportan como un único ordenador [9].

La visión general de una arquitectura basada en un cluster es la siguiente [10]:

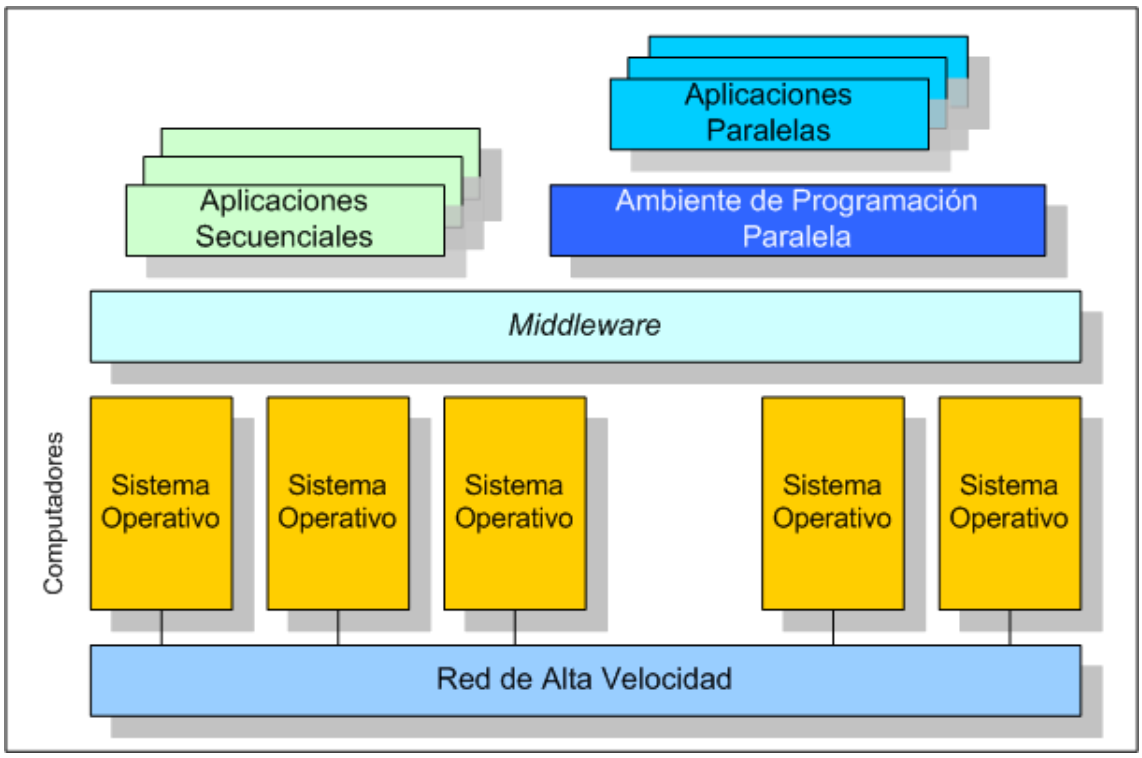


Figura 6: Visión general de una arquitectura cluster incluyendo todos sus componentes

Como se observa en la Figura 6 los componentes de un cluster son los siguientes:

- Nodos: Ordenadores empleados para realizar el cluster.
- Sistemas Operativos: De uso sencillo, permiten múltiples usuarios y procesos en paralelo.
- Conexiones de red: Los nodos del cluster se conectan a través de una red de comunicaciones como puede ser un cable Ethernet o fibra óptica.
- Middleware: Actúa entre el sistema operativo y las aplicaciones ofreciendo la posibilidad de manejo desde un solo ordenador. Además permite un mantenimiento y una escalabilidad sencillas.

2.1.3. Infraestructura

2.1.3.1. Apache Hadoop

2.1.3.1.1. Introducción

Apache Hadoop [2] es un framework de código abierto basado en Java que permite el procesamiento distribuido de grandes conjuntos de datos a través de un cluster de ordenadores utilizando modelos de programación simples. Está diseñado para para escalar desde un solo nodo hasta cientos de máquinas, cada una de ellas ofreciendo almacenamiento y computación locales. Está diseñado para detectar errores en la capa de aplicación. Siendo tolerante a fallos.

Apache Hadoop presenta cuatro módulos principales:

- **Hadoop Common:** Librerías de Java que soportan el resto de módulos
- **Hadoop Distributed File System (HDFS):** Se trata de un sistema de archivos distribuido que provee de acceso a los datos con un rendimiento elevado.
- **Hadoop Yarn:** Es un framework encargado de la planificación de tareas y de la gestión de los recursos del cluster.

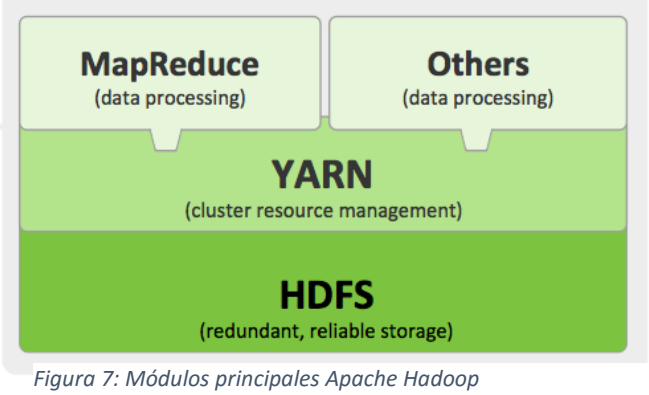


Figura 7: Módulos principales Apache Hadoop

- **Hadoop Mapreduce:** Sistema basado en Yarn para el procesamiento paralelo de grandes conjuntos de datos.

2.1.3.1.2. Funcionamiento

Procesos (daemons)

En la versión de Mapreduce utilizada a lo largo de este proyecto, la versión V2, la lista de procesos que deben ejecutarse para un correcto funcionamiento de Hadoop es la mostrada en la Figura 8 [11].

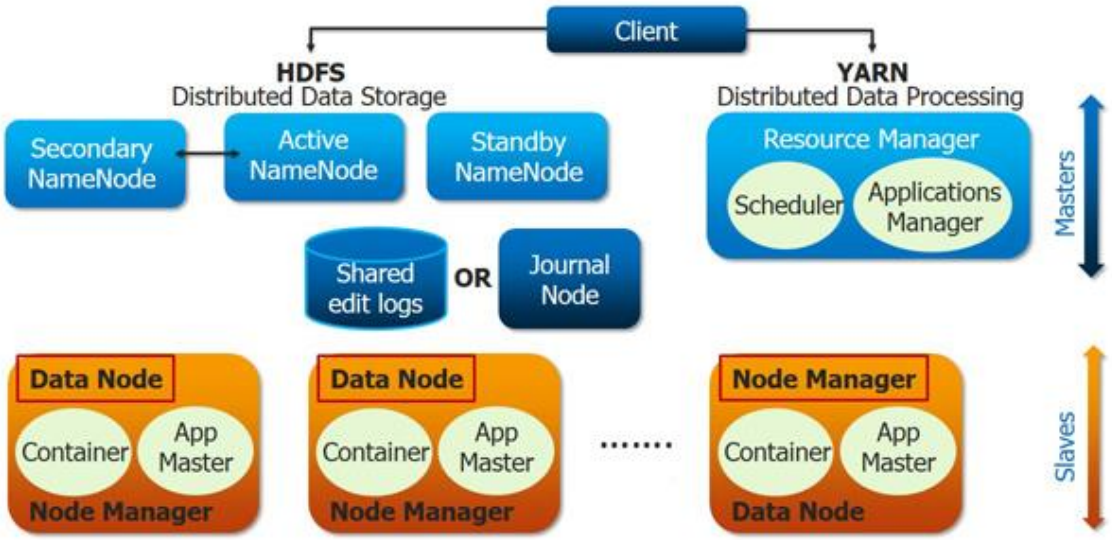


Figura 8: Procesos (daemons) ejecutados en Apache Hadoop

- **Namenode:** Es la pieza central del sistema de ficheros HDFS, se ocupa del mantenimiento del árbol de ficheros del sistema, asegurando el lugar dónde se encuentra cada fichero pero sin llegar a almacenarlo.
- **Secondary Namenode:** Se encarga del mantenimiento del Namenode y se encuentra en una máquina diferente.
- **Datanode:** Almacena los datos en el HDFS. Un sistema funcional presenta varios datanodes con función de replicación a lo largo de ellos.
- **Resource Manager:** Gestiona todos los recursos del cluster y permite controlar las aplicaciones distribuidas que se ejecutan en Yarn.
- **Node Manager**: Recibe instrucciones del Resource manager y gestiona los recursos disponibles de un nodo.

HDFS

HDFS (Hadoop distributed file system) es un sistema de ficheros con una tolerancia a errores muy alta, diseñado para funcionar en hardware de bajo coste. Debido a su relevancia, es ya un subproyecto dentro de Hadoop.

Como responsable del almacenamiento de los datos en el sistema, estos se dividen en bloques de 64 Mb (por defecto) y son enviados a varios nodos del cluster dependiendo del factor de replicación establecido.

Cabe destacar que HDFS está optimizado para la lectura de cantidades de datos muy grandes con el objetivo de reducir al máximo la latencia. La estructura interna de HDFS es la mostrada en la Figura 9 [12].

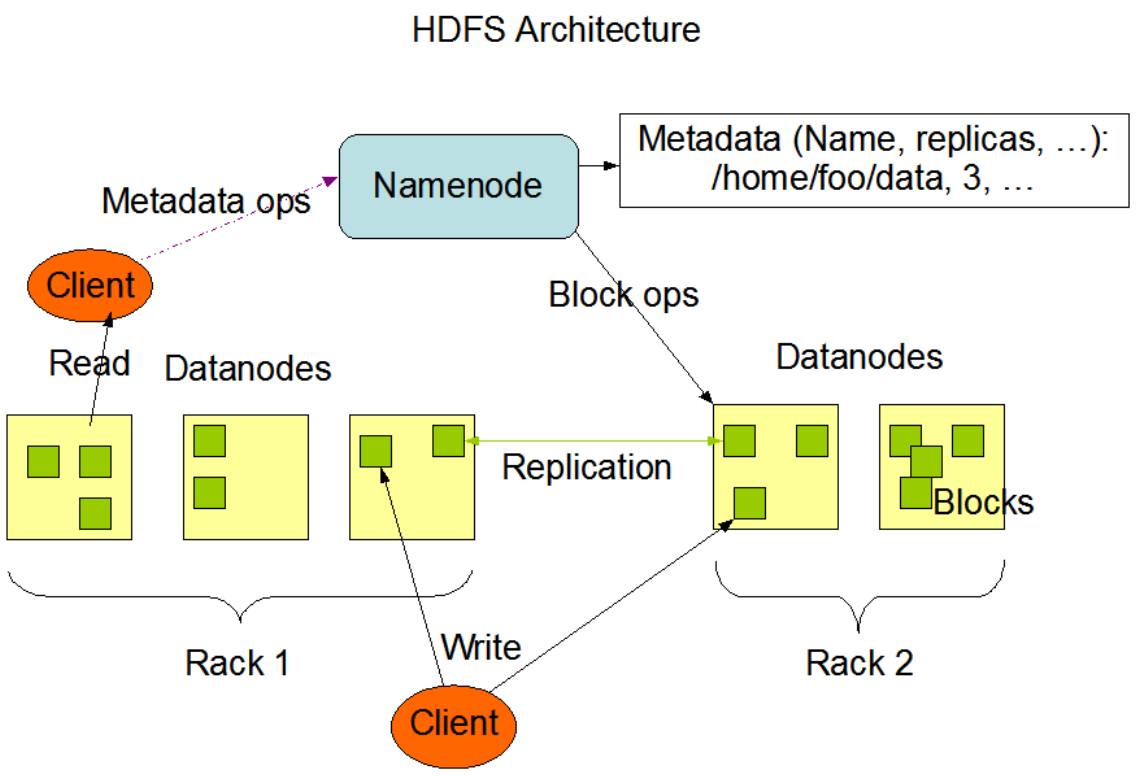


Figura 9: Visión general arquitectura HDFS

El procedimiento de lectura y escritura de los diferentes bloques en HDFS es el siguiente:

- Escritura de bloques:

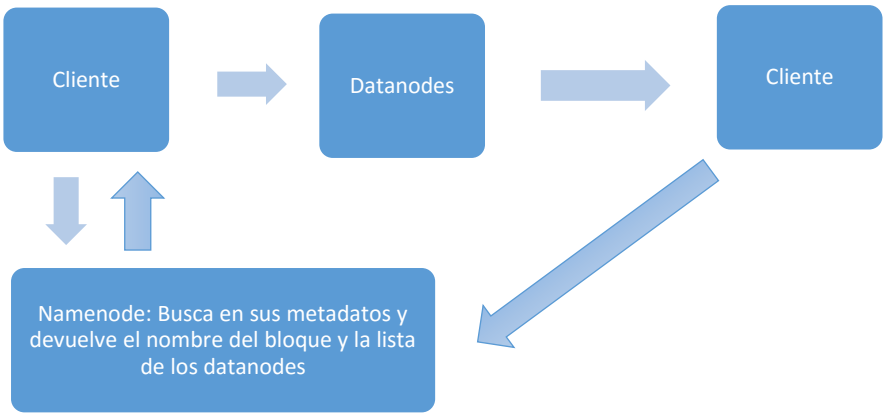


Figura 10: Proceso de escritura en HDFS

- Lectura de bloques:

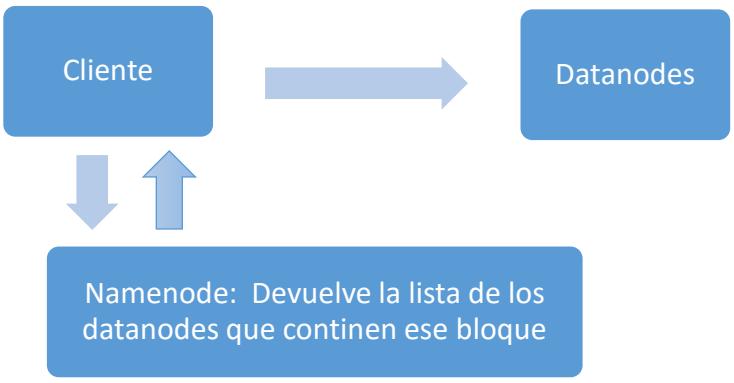


Figura 11: Proceso de lectura en HDFS

Mapreduce

Mapreduce es un framework software que permite una escritura sencilla de aplicaciones para el procesamiento de grandes cantidades de datos en paralelo en clusteres de gran tamaño.

Mapreduce divide los datos de entradas en bloques independientes que son procesados por los procesos de mapeo de una manera paralela. Posteriormente se reparten esos bloques mapeados entre las tareas de 'reduce'.

El objetivo del mapeo y la reducción es el de obtener un procesamiento de los datos más rápido a partir de la división de los mismos para ser procesados en paralelo por distintos nodos.

El funcionamiento de Mapreduce se ilustra en la siguiente figura [13] :

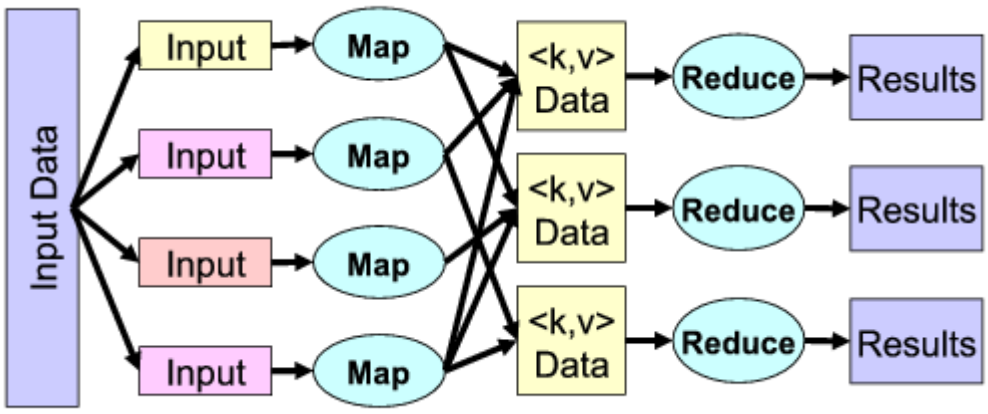


Figura 12: Esquema general del funcionamiento de Mapreduce

2.1.3.2. Apache Hive

2.1.3.2.1. Introducción

Apache Hive [3] es un software de almacenamiento de datos que facilita la lectura, escritura y tratamiento de datos almacenados en HDFS utilizando un lenguaje similar a SQL, llamado HiveQL. Proporciona además una herramienta de línea de comandos para facilitar su usabilidad. Es un software que trabaja directamente encima de Hadoop y por tanto requiere de él para funcionar. Las principales características de Hive son las siguientes:

- Herramientas para facilitar el acceso a los datos a través de SQL.
- Soporte para diferentes estructuras de datos.
- Acceso a los ficheros almacenados en HDFS.

2.1.3.2.2. HiveQl

El lenguaje empleado para realizar instrucciones es muy similar a SQL pero con la excepción de no poder modificar tablas creadas, esto se realiza para garantizar un buen rendimiento dentro de Hadoop ya que las operaciones de modificación son muy costosas.

2.1.4. Mercados Financieros

2.1.4.1. Introducción

Un mercado financiero es aquel lugar (físico o electrónico) que permite la negociación de activos e instrumentos financieros, poniendo en contacto a los oferentes y a los demandantes de dichos productos fijando un precio público de venta para su posible adquisición. El funcionamiento de los mercados se rige por la ley de la oferta y la demanda.

El papel de los mercados financieros en la economía mundial actual es muy relevante, favoreciendo el desarrollo económico y empresarial y actuando como instrumento de financiación de las empresas gracias a la canalización de los flujos monetarios.

2.1.4.2. Mercados financieros destacados

Dentro de la gran cantidad de mercados bursátiles existentes, hay una serie de ellos que afectan el normal funcionamiento del resto, teniendo un impacto directo sobre ellos, dependiendo de su tendencia. De esta manera, y por proximidad el mercado que más

afecta a la economía española es el Ibex 35, formada por las 35 empresas con más capital del país, entre ellas los grandes bancos como el BBVA o el Santander, empresas textiles como Inditex o grandes constructoras como FCC o Ferrovial, así como eléctricas como Iberdrola. Además, los mercados financieros internacionales que afectan al Ibex35 son:

- Nikkei: Se trata del mercado Japonés, formado por 225 valores. Su gran economía y sus continuos periodos de recesión hacen que este mercado juegue un papel relevante en la economía mundial.
- Nasdaq: Es el principal mercado tecnológico mundial, interesante debido a la participación de las grandes empresas tecnológicas. Incluye 100 valores, que son las 100 empresas de hardware, software y comunicaciones más importantes.
- Ftse: Compuesta por los 100 valores más importantes de la bolsa londinense. La importancia de la economía británica en la eurozona hacen de este un mercado a seguir.

2.1.4.3. La importancia del análisis de datos ágil y eficaz en el mundo económico

Al igual que en el resto de sectores, el aumento del número de datos y de mercados emergentes hacen que aparezca una necesidad de analizar esos datos de una manera eficaz.

Al tratarse de un mundo muy dinámico, la volatilidad de los datos es muy elevado, por lo que conseguir herramientas de análisis y de predicción, fiables y rápidas otorga una ventaja competitiva grande.

Los grandes bancos, los fondos de inversión e incluso las administraciones públicas demandan sistemas que les permitan analizar esos datos que con herramientas tradicionales no son capaces, exigiendo las dos capacidades de los sistemas Big Data, rapidez y capacidad de grandes conjuntos de datos otorgándoles a estos un sentido.

2.2. Marco regulador

Pese a que el paradigma Big Data es relativamente reciente, ya existen regulaciones tanto técnicas a cerca de la propia tecnología como relativa a los datos que el Big-data procesa, por ello hay que tener en cuenta ambas, aunque se refieran a ámbitos diferentes. A continuación se hace un análisis de los dos tipos de normativas aplicables al entorno Big-data.

2.2.1. Marco regulador técnico

La regulación técnica viene determinada por diferentes organizaciones y estamentos [14] que aplican a temas concretos dentro del Big Data a través de estándares, los existentes en la actualidad son las siguientes:

Consorcio/Estándar	Área	Tareas principales
ISO/IEC JTC 1/SC 32	Tratamiento e intercambio de datos. Bases de datos, lenguajes base de datos, tratamiento multimedia y metadatos, comercio electrónico	Estándares de comercio electrónico, repositorios de metadatos, SQL
ISO/IEC JTC 1/SC 38	Estandarización para plataformas y servicios de aplicaciones distribuidas interoperables, incluyendo servicios web, SOA y computación en la nube	Mantenimiento de Interfaces de datos en la nube, virtualización de formato abierto e interoperabilidad entre servicios web
ITU-T SG13	Computación distribuida para Big Data	Computación distribuida basada en requisitos Big-data, capacidades y ejemplos de uso
Organization for the Advancement of Structured Information Standards	Acceso e intercambio de información	Conjunto de protocolos para interactuar con datos estructurados, estándares para seguridad, computación en la nube, servicios web, SOA, redes inteligentes y otras aéreas
ON	Puntos de referencia para sistemas Big Data	Especificaciones para TPC Express y Benchmark para el sistema Hadoop y su entorno
TM Forum	Apoyo a empresas, proveedores de servicios y proveedores para una continua transformación en busca de éxito en la economía digital	Compartir experiencia para solucionar retos de negocio críticos, optimización de procesos de negocio, análisis Big Data, mantenimiento de la nube y ciber seguridad.

Tabla 1: Estándares y organizaciones regulación Big Data

2.2.2. Privacidad de datos

Además De unos estándares técnicos, existen también regulaciones en cuánto a los datos que se utilizan en el entorno Hadoop. Estos datos, en muchas ocasiones pueden ser datos personales, los cuáles no pueden ser tratados sin las medidas oportunas.

A nivel nacional aplica la ley orgánica Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal, la cual "*tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar*." [15]

A nivel Europeo el referente en privacidad de datos aplicables a Big Data es el Grupo de trabajo del artículo 29 de protección de datos [16], el cual es un órgano consultivo independiente integrado por las Autoridades de Protección de Datos de todos los Estados miembros, el Supervisor Europeo de Protección de Datos y la Comisión Europea - que realiza funciones de secretariado-. La Agencia Española de Protección de Datos forma parte del mismo desde su inicio, en febrero de 1997.

En el dictamen 05/2014 [17] emitido por el grupo, se describen técnicas de anonimización de datos, con el objetivo de que los datos personales recogidos dejen de estar vinculados a una persona para que no se pueda proceder a su identificación una vez que han sido procesados. Hay tres características a tener en cuenta en este proceso de anonimización:

- Inferencia: Posibilidad de identificar un atributo a partir de otro conjunto de atributos.
- Vinculación: Posibilidad de vincular dos registros de una persona estando en la misma base de datos o en diferentes bases de datos.
- Singularización: Extracción de un registro que identifique a una persona.

A partir de las técnicas de anonimización se intentan evitar las tres características anteriores.

Dichas técnicas, por recomendación y análisis del Grupo 29, pueden ser las siguientes:

1. Aleatorización: Se utilizan para eliminar la relación existente entre el dato y la persona protegiendo contra ataques o riesgos de inferencia.
    - Adición de ruido
    - Permutación
    - Privacidad diferencial

2. Generalización: A diferencia de la aleatorización es menos exhaustiva, generalizando ciertos atributos de los registros modificando los órdenes de magnitud.
    - Agregación y anonimato K
    - Diversidad l, proximidad t
3. Pseudo-anonimización: El uso por si solo de esta técnica no garantiza la privacidad de los datos, por lo que debe ser usada en combinación con otras técnicas. Consiste en el intercambio de un registro dentro de un conjunto de datos.
    - Cifrado con clave secreta
    - Función Hash
    - Función con clave almacenada
    - Cifrado determinista con borrado de clave
    - Descomposición en tokens

BLOQUE III

Trabajo realizado

3. Diseño del sistema

3.1. Introducción

Tras haber analizado las tecnologías Hadoop y Hive que vamos a emplear, procedemos al diseño del sistema completo, indicando los bloques funcionales del mismo y la interacción entre ellos. Por último se detalla el diseño del sistema presentando varias alternativas, cada una con una configuración distinta.

3.2. Bloques Funcionales

El sistema Big Data planteado presenta tres bloques funcionales claramente definidos. El primero de ellos es el bloque encargado de la obtención de datos, en dónde el usuario del sistema recaba los datos a procesar, posteriormente nos encontramos con el bloque encargado de introducir los datos en el sistema. Por último están presentes el bloque de tratamiento de datos y el de obtención de resultados, actuando por separado. Dichos bloques se exponen a continuación:

- **Obtención de datos**: El primer paso a ejecutar en el sistema es el de la obtención de los datos, en este caso, procedentes de mercados financieros, y más concretamente de Yahoo finance [1], que nos provee de datos actualizados constantemente, siendo una fuente fiable de datos. Los datos procedentes de esta fuente presentan la misma estructura entre ellos, pudiendo variar de mercado financiero sin afectar al funcionamiento del sistema. No se puede garantizar el correcto funcionamiento del sistema con datos procedentes de otras fuentes sin antes haber hecho alguna pequeña modificación si la estructura de los mismos varía.

  El documento de datos obtenido desde Yahoo Finance presenta un formato csv, compatible con nuestro sistema Hadoop y Hive.

- **Introducción de datos**: Una vez se han obtenido los ficheros de datos, se introducen los datos en el sistema a través de Hive, el cual se encarga de distribuir los mismos a través de HDFS. Dichos datos consisten en registros de días, con información sobre el valor de apertura, cierre y volumen del mercado financiero.

- **Tratamiento de datos**: Con los datos introducidos en el sistema se puede realizar el análisis de los mismos a partir de los requisitos establecidos, que se explicarán en el apartado 4. Para dicho análisis, el cual busca obtener un resultado a partir

de los datos procesados, se utiliza instrucciones HiveQL recogidas en algoritmos para facilitar su utilización.

- **Obtención de resultados**: A partir del análisis de datos realizado se obtiene una serie de resultados, dichos resultados se almacenan en tablas internas del sistema Hadoop, que puede ser extraídas para ser procesados por herramientas de visualización de datos, pero además, se obtienen resultados en forma de porcentajes a través de la consola de comandos.

A continuación se muestra el esquema general de nuestro sistema Big Data:

Figura 13: Esquema general del sistema Big Data incluyendo los bloques funcionales

Todos los elementos del sistema interactúan entre sí, creando una serie de dependencias que hacen que el sistema funcione de la forma deseada.

3.3. Arquitectura del sistema

Diseñamos la arquitectura del sistema ofreciendo dos posibles soluciones con el objetivo de crear un sistema polivalente que se ajuste a diferentes escenarios. Para ello se utilizan dos tipos de configuraciones que el propio sistema Hadoop nos permite, en primer lugar una configuración pseudo-distribuida y posteriormente una configuración multinodo.

Además, dentro de la configuración mutinodo se plantean dos arquitecturas con diferentes comunicaciones y diferentes números de nodos.

El objetivo de plantear diferentes escenarios y configuraciones es el de evaluar cómo afectan los diferentes componentes del sistema en la eficiencia del mismo, tratando de obtener soluciones diferentes para conjuntos de requisitos que difieran en exigencia.

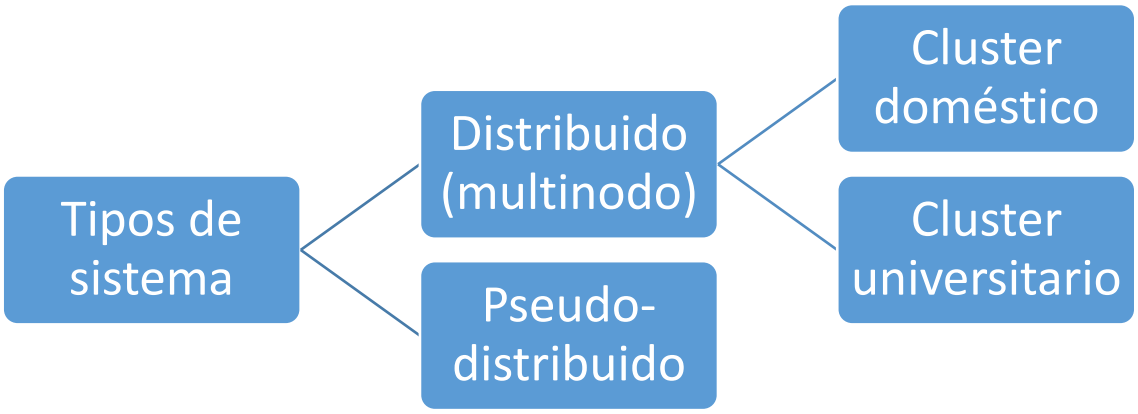


Figura 14: Esquema de los tipos de sistema Big Data planteados

3.4.1. Sistema Pseudo-Distribuido

En este tipo de sistema, tanto el maestro como el esclavo se encuentran en la misma máquina, posibilitando así la implementación de un sistema Hadoop completo en un solo nodo.

La estructura global de esta configuración es la mostrada en la Figura 15, en ella se puede ver lo destacado anteriormente, que todos los procesos se ejecutan en el mismo ordenador, al contener tanto al maestro como al esclavo.

Dentro del rol de maestro, este ejecuta el namenode, el secondary namenode y el resourcemanager, mientras que el esclavo ejecuta el datanode y el nodemanager.

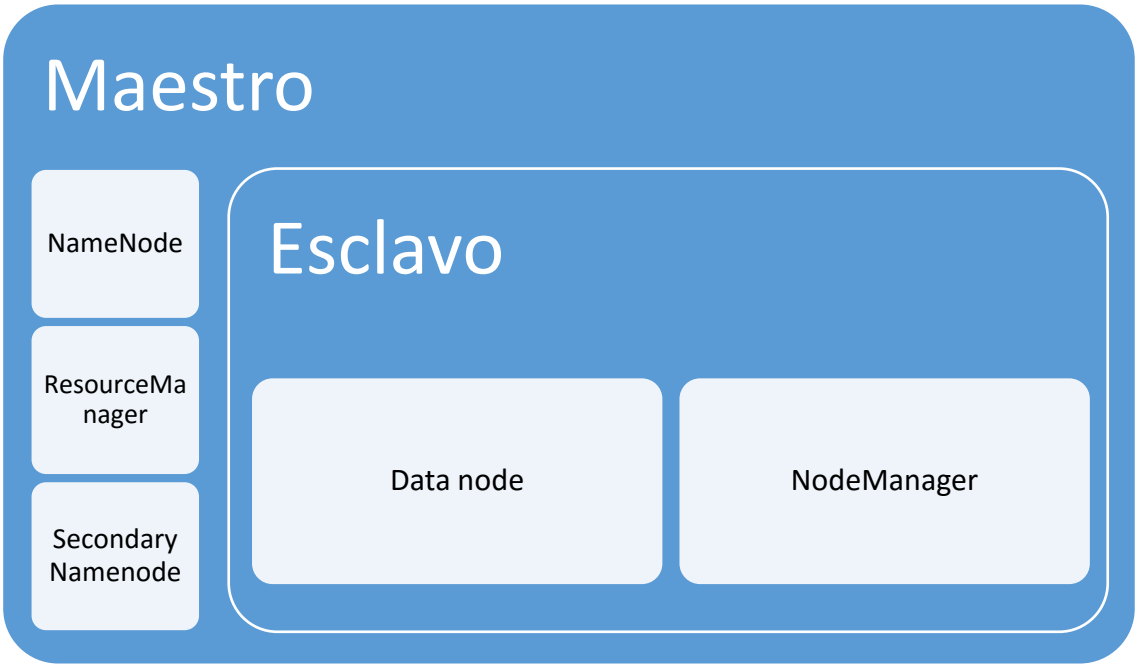


Figura 15: Visión general del sistema pseudo-distribuido incluyendo los procesos que en él participan

Esta configuración tiene tanto ventajas como desventajas, las cuales se demostrarán en el apartado 6. Dichas ventajas y desventajas son las siguientes:

Ventajas

-Facilidad de instalación: La instalación se realiza en una sola máquina, por lo que es más sencillo y rápido de instalar que un cluster.

-Mantenimiento: El mantenimiento es menos costoso al tratarse de una sola máquina.

Desventajas

-Limitación computacional: La capacidad de computación de una sola máquina es evidentemente, mucho menor que la de un cluster, llegando a ser incapaz de procesar archivos de datos muy grandes.

3.4.2. Sistema Distribuido

El sistema distribuido es el sistema más comúnmente utilizado debido a su capacidad de procesamiento de datos con eficiencias elevadas dependiendo del número de nodos.

Cómo solución al problema planteado presentamos dos soluciones que difieren en cuánto al entorno, número de máquinas y métodos de conexión. Comenzaremos por una configuración menos profesional, un cluster doméstico y posteriormente pasaremos a una configuración más profesional, un cluster universitario, aprovechando los recursos disponibles en la universidad Carlos III.

3.4.2.2. Cluster doméstico

El objetivo de esta configuración es el de implementar un sistema Hadoop en un entorno doméstico, aprovechando los recursos disponibles. En este caso, nos encontramos con una limitación de máquinas, es por ello que posteriormente se traslada el sistema a un entorno universitario dónde el número de máquinas es mayor.

El número de nodos implementado es de tres sin contar con el nodo maestro. El esquema de la arquitectura montada es el siguiente:

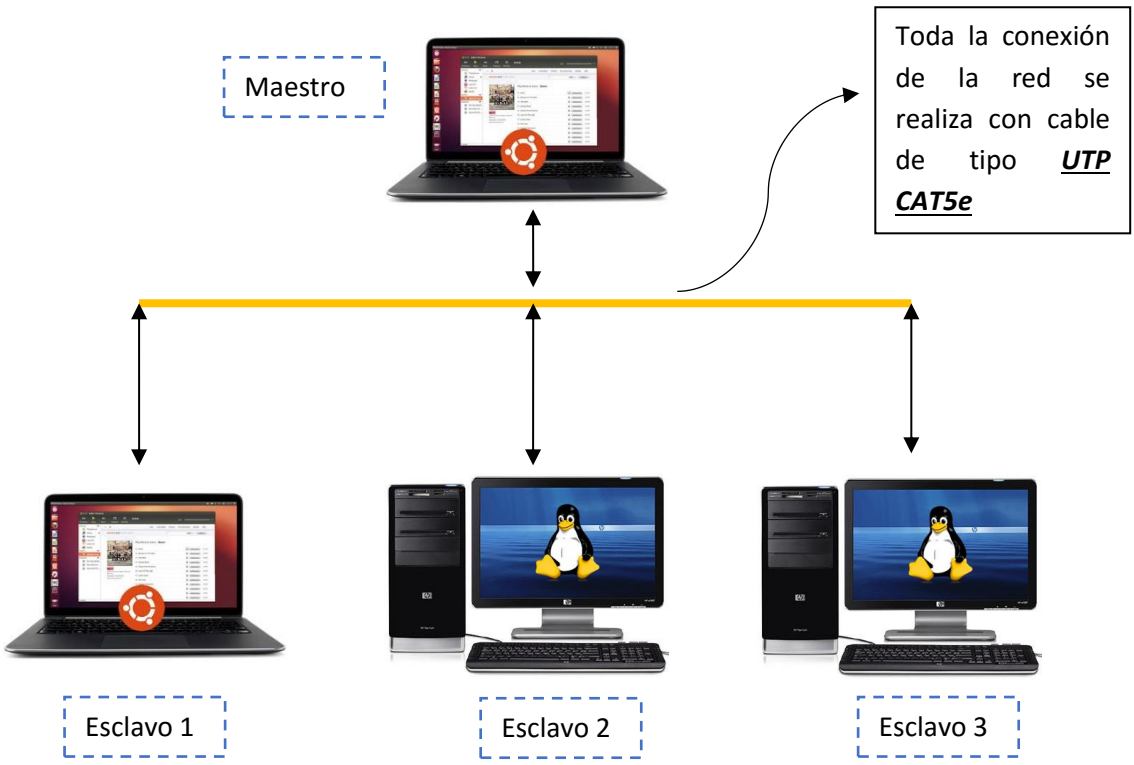


Figura 16: Arquitectura cluster doméstico diseñada

Tanto el maestro como el esclavo consisten en un ordenador portátil, mientras que los esclavos 2 y 3 consisten en un ordenador de sobremesa, todos con Ubuntu 15.10 instalado. Los detalles sobre la instalación y las características de cada máquina se encuentran en los apartados 4.2 y 6.2.

El funcionamiento interno de Hadoop en esta ocasión es diferente al de la configuración distribuida. Al aumentar el número de nodos cabe la posibilidad de replicar los datos en algunos de ellos, gracias al factor de replicación, tal y como se explicó en el apartado 2.1.3.1.2, en esta ocasión y siguiendo las recomendaciones de la documentación oficial, ha sido establecido en 3. Esto quiere decir que los bloques de datos serán replicados a lo largo de 3 máquinas, teniendo varios procesos datanode activos, uno por cada máquina.

Dicha configuración se puede observar en la siguiente figura:

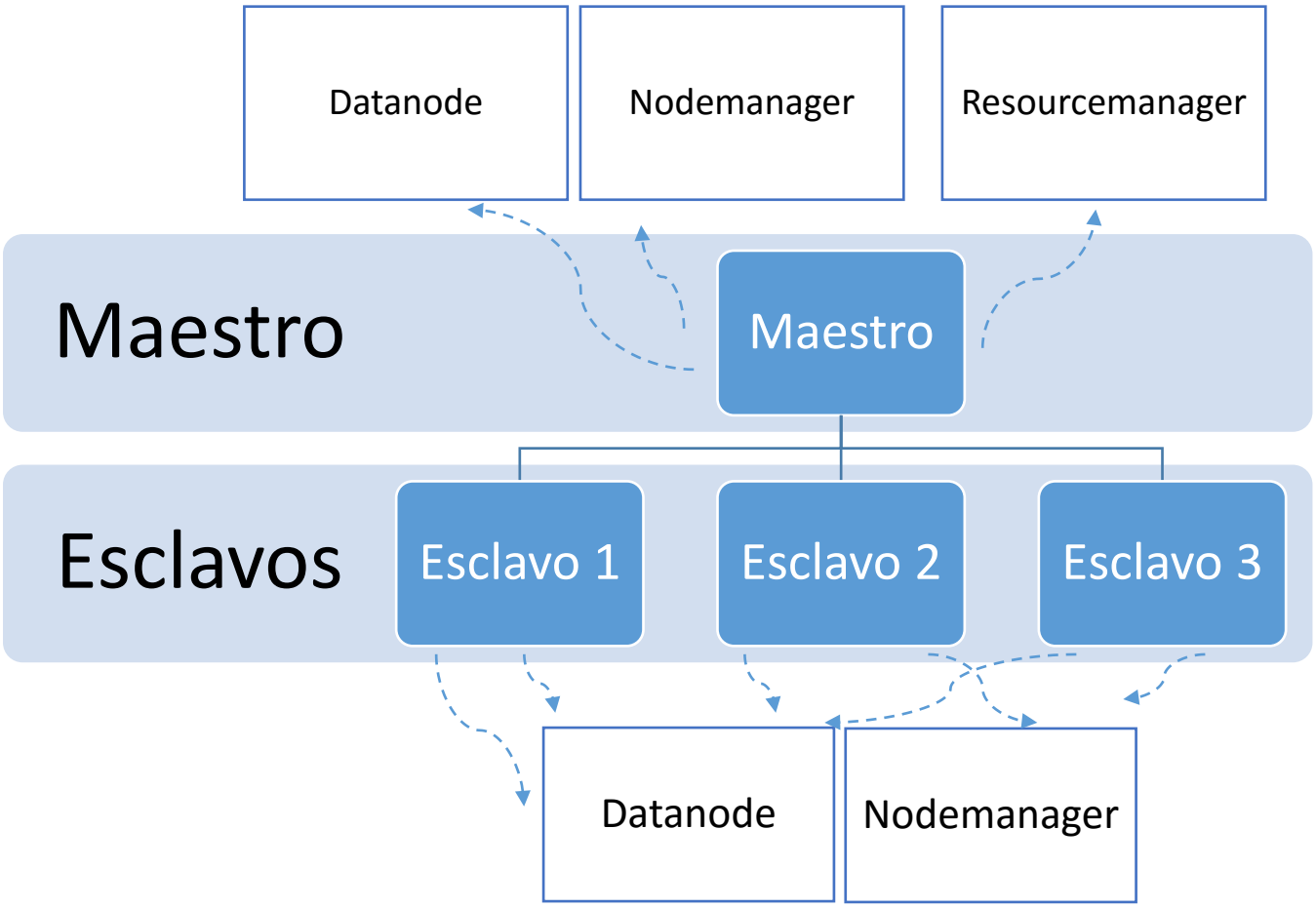


Figura 17: Configuración aplicada al cluster domestico incluyendo los procesos Hadoop

Al igual que la configuración pseudo distribuida, esta configuración también cuenta con ventajas y desventajas, como se muestra a continuación:

Ventajas

-Capacidad de computación: La capacidad de computación es mayor que la de una sola máquina.

Desventajas

-Conexión: Las conexiones entre máquinas se realizan con cable UTP, limitando la comunicación entre nodos en los procesos de Mapreduce.

3.4.2.2. Cluster universitario

Una vez implementado el sistema Hadoop en un entorno doméstico nos disponemos ahora a desplegarlo en un entorno profesional, en dónde todas las máquinas que actuarán como esclavos y el maestro tienen las mismas especificaciones.

Además, a la hora de implementarlo nos servirá para conocer la escalabilidad de nuestra configuración del cluster doméstico, ya que se trata del mismo sistema con un número de máquinas mayor.

El esquema de esta arquitectura es el siguiente:

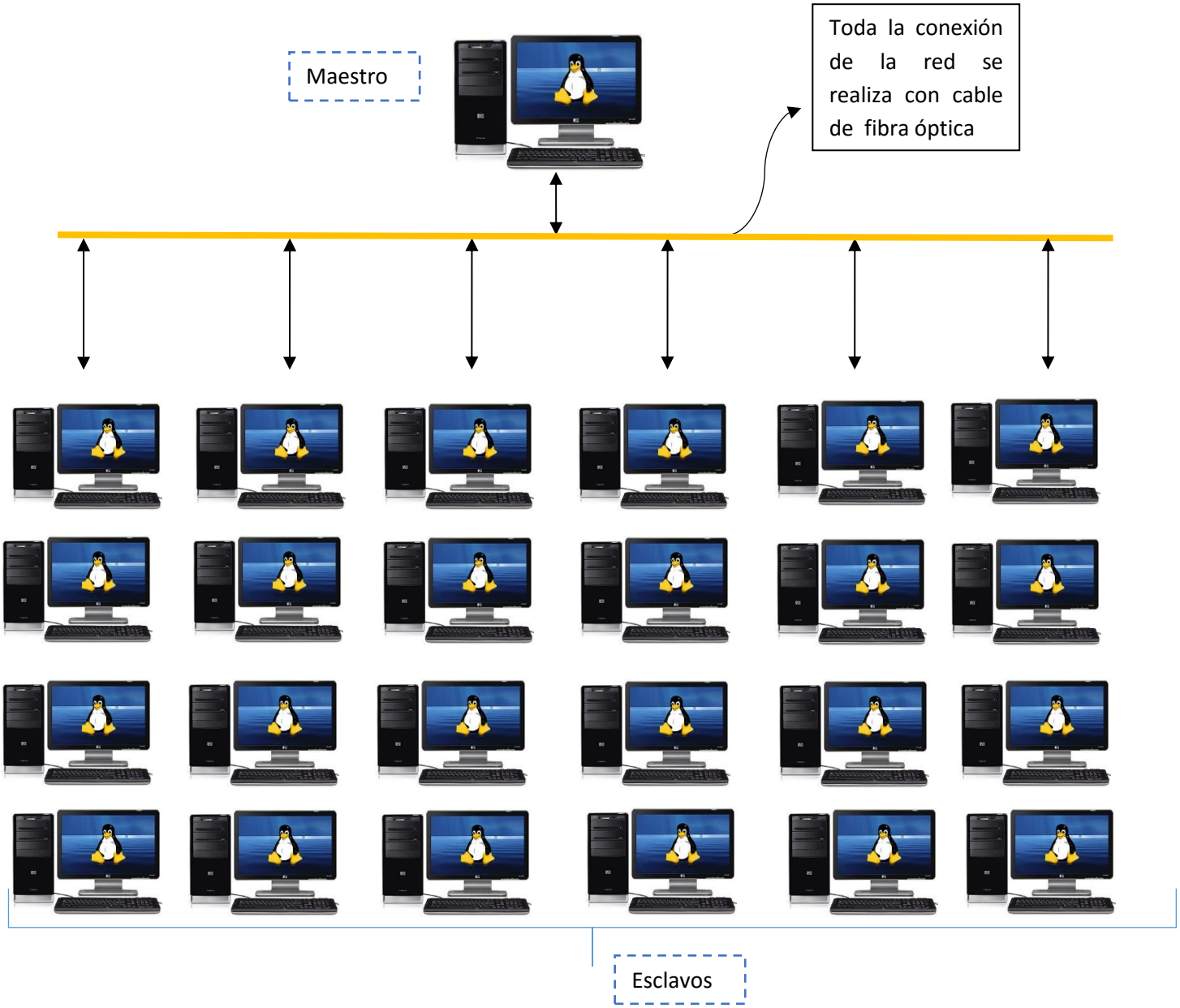


Figura 18: Arquitectura cluster universitario diseñada

Se trata por tanto de una configuración con un máximo de 24 nodos, configurable de manera sencilla para disponer de los nodos deseados en cualquier momento.

En esta ocasión la configuración propuesta no cuenta con factor de replicación, ya que al tratarse de un sistema estable, no es necesaria la replicación de los datos a lo largo de las máquinas. Además, de esta manera evitamos el tráfico de datos que se generaría entre máquinas, aumentando el rendimiento de la infraestructura. Dicha configuración se puede observar en la siguiente figura:

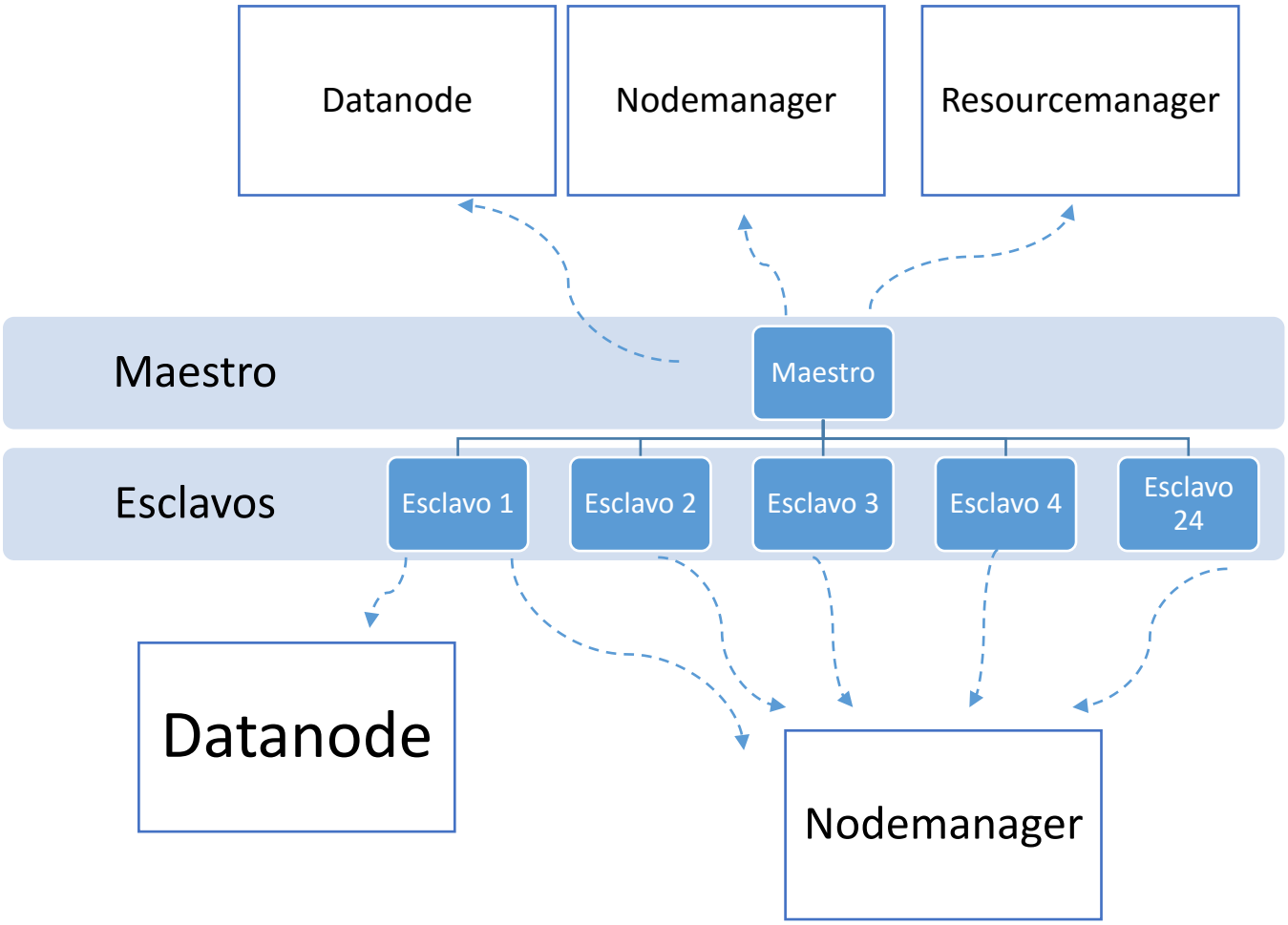


Figura 19: Configuración aplicada al cluster universitario incluyendo los procesos Hadoop

Al igual que el cluster doméstico, esta configuración también cuenta con ventajas y desventajas, como se muestra a continuación:

Ventajas

-Capacidad de computación: La capacidad de computación es mucho mayor que la del cluster doméstico, y que por supuesto que la de la configuración pseudo-distribuida.

-Conexión: Las conexiones entre máquinas se realizan con fibra óptica, mejorando la comunicación entre nodos en los procesos de Mapreduce.

Desventajas

-Coste: El coste es elevado debido al gran número de máquinas necesarias para implementar la infraestructura.

-Capacidad de disco: La capacidad de almacenamiento, está limitada a 2,8 Gb en los laboratorios de telemática de la universidad Carlos III.

4. Implementación

4.1. Introducción

En este apartado se detallan todos los aspectos de la infraestructura Big Data que se planteó en el apartado 3. Diseño del sistema.

Cómo parte de la implementación de la arquitectura, se plantean dos posibles soluciones, aquella en la que nodo y esclavo se ejecutan en la misma máquina (solución pseudo-distribuida), y aquella en la que el maestro se ejecuta en una máquina diferente que los esclavos (solución multinodo). Se decide emplear estas dos soluciones de cara a ver el rendimiento de cada una de ellas, para así poder obtener una visión detallada de lo que nos pueden aportar a la hora de tener que realizar un análisis de millones de datos.

En este apartado 4 del proyecto se describirán todos los pasos necesarios para obtener un sistema completo y funcional para el análisis de grandes cantidades de datos. Se detallarán las instalaciones necesarias; Linux, Hadoop, Hive, los procedimientos para la obtención, transformación y análisis mediante scripts de los datos procedentes de los diferentes mercados financieros escogidos que serán objeto de nuestro estudio, y por último, se expondrán los resultados obtenidos mediante diferentes técnicas de visualización de datos.

4.2. Preparación del entorno de trabajo

Como se expuso anteriormente (véase apartado Apache Hadoop), para poder ejecutar Hadoop necesitamos instalarlo en un sistema operativo. La solución escogida ha sido Linux, y concretamente la distribución Ubuntu [18], esta elección atiende a razones de soporte, ya que la comunidad existente en torno a Ubuntu y Hadoop es mucho mayor que para Windows, además, gran parte de la documentación oficial de Hadoop [19] se encuentra disponible para Linux y por último, se trata de un sistema operativo open-source, que no requiere de ningún tipo de licencia, lo que abarata los costes de la infraestructura. Por tanto, el haber elegido la combinación Linux y Ubuntu nos facilita la implementación y el desarrollo del análisis de datos.

A su vez, para poder ejecutar Hive, se necesita haber instalado Hadoop.

En los apartados siguientes se explica cómo instalar las tres herramientas necesarias para implementar la infraestructura, comenzando por la instalación de Linux y acabando con Hadoop y Hive sobre Ubuntu (Linux).

4.2.1. Instalación Linux

Como se ha comentado anteriormente en este documento, la distribución Linux escogida para la realización del sistema es Ubuntu.

4.2.1.1. Requisitos previos

Para poder instalar Linux en una máquina, hace falta cumplir una serie de requisitos básicos en forma de hardware, los mostrados a continuación, son los requisitos mínimos para que el sistema funcione correctamente:

- Procesador dual-core trabajando a una frecuencia de 2 GHz
- 2 GB de memoria ram
- 25 GB libres en el disco duro
- Lector de DVD o puerto Usb para poder instalar el sistema
- Recomendado: Acceso a internet

4.2.1.2. Descarga del Sistema operativo

Accedemos a la web de descargas de Ubuntu [20] y descargamos la última versión de 64-bits estable disponible. La versión con la que trabajaremos en la solución propuesta en este documento es la 15.10, ya que era la última versión estable disponible en el momento del inicio del proyecto.

4.2.1.3. Creación del soporte de instalación

Para instalar Ubuntu 15.10 en nuestro ordenador creamos una memoria usb bootable[1]. Para ello utilizamos el programa Rufus [21], disponible únicamente para Windows. El tamaño mínimo exigido de la memoria usb para poder crear el soporte de instalación correctamente debe ser de al menos 2gb.

Una vez que tenemos descargado tanto la versión de Ubuntu que deseamos como el programa Rufus, realizamos el usb bootable:

1. Seleccionamos en la pestaña “dispositivo” el usb que queremos convertir en bootable.

2. Seleccionamos en el icono a la derecha del desplegable “FreeDOS” el fichero de la versión de Ubuntu escogida en formato ISO descargado anteriormente.

[1] Dispositivo de almacenamiento que nos permite arrancar un proceso de instalación en un pc a través de él.

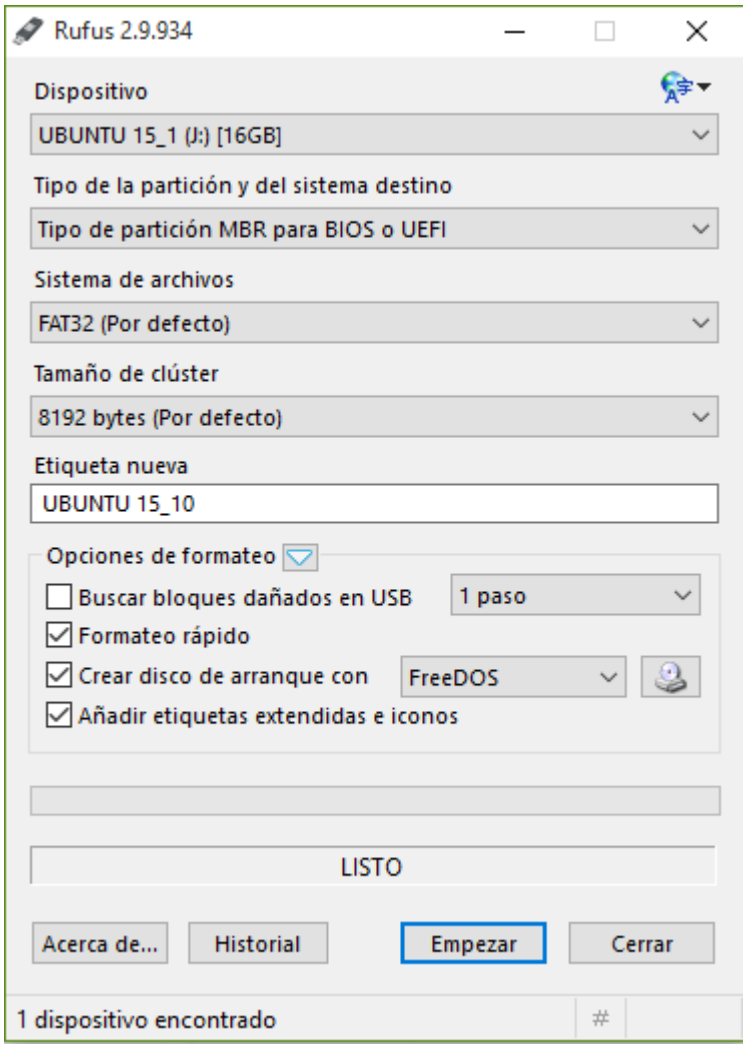


Figura 20: Creación de un usb bootable a través del programa Rufus

4.2.1.4. Instalación de Ubuntu

Por último, para instalar Ubuntu en el pc elegido, se utiliza el usb creado en el apartado anterior.

Para poder arrancar la instalación desde el usb es necesario modificar el orden de arranque de los dispositivos en la Bios del Pc, seleccionando como método de arranque prioritario el usb, por encima del disco duro.

Una vez arrancado, la instalación se realiza siguiendo los pasos que aparecen en pantalla.

Cabe destacar que el usb creado servirá como soporte de instalación para cada uno de las máquinas que queramos añadir a nuestra infraestructura Big Data, ya que todas ellas deben utilizar la misma versión de Ubuntu, para evitar problemas de compatibilidades.

4.2.2. Instalación Hadoop

Como parte de nuestra implementación, necesitamos realizar dos tipos de instalaciones de Hadoop, por una parte la instalación basada en la configuración pseudo-distribuida en dónde el único nodo se encuentra en la misma máquina que el maestro y por otro lado la instalación basada en configuración multinodo, en dónde cada nodo se encuentra en una máquina diferente. A continuación se expone el procedimiento para realizar cada una de las instalaciones citadas [22] [23]:

4.2.2.1. Prerrequisitos

Los requisitos previos [22] antes de proceder a realizar cualquiera de las dos configuraciones explicadas son los siguientes:

- Tener instalado Linux instalado (véase apartado 4.2.1 Instalación Linux)
- Tener instalado Java en Linux
- Tener instalado rsync en Linux
- Existencia de un usuario y grupo común en todas las máquinas
- Tener instalado ssh en Linux

4.2.2.1.1. Instalación Java

Es fundamental y requisito indispensable Java, ya que Hadoop está soportado directamente encima.

Para poder realizar la instalación de Java nos dirigimos a un repositorio Linux que nos proporcione actualizaciones recientes y fiables, en este caso nos dirigimos a WebUpd8 [24]

Comandos a ejecutar en la consola de comandos de Linux para la instalación:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Figura 21: Comandos instalación Java en Linux

4.2.2.1.2. Instalación rsync

Comandos a ejecutar en la consola de comandos de Linux para la instalación:

```
sudo apt-get install rsync
```

Figura 22: Comandos instalación Rsync en Linux

4.2.2.1.3. Creación de usuario y grupo común

Es necesario la existencia del mismo usuario y grupo en cada una de las máquinas de la infraestructura para que las operaciones de HDFS y MapReduce que son bilaterales se puedan ejecutar sin ningún problema de permisos en Linux.

Comandos a ejecutar en la consola de comandos de Linux para la instalación:

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop hduser
```

Figura 23: Comandos creación usuario y grupo en Linux

Es muy importante acceder a este usuario la primera vez que se arranca Linux al comienzo de cada sesión, ejecutando todas las órdenes bajo el usuario hduser creado dentro del usuario Hadoop.

4.2.2.1.4. Instalación ssh

Comandos a ejecutar en la consola de comandos de Linux para la instalación:

```
sudo apt-get install openssh-server
```

Figura 24: Comandos instalación Open-SSH en Linux

Una vez instalado ssh hay que realizar la configuración del mismo en cada una de las máquinas, con el objetivo de compartir los certificados que ssh crea y así poder establecer las comunicaciones entre todos los nodos.

Comandos a ejecutar en la consola de comandos de Linux para la configuración ssh:

```
sudo su hduser
ssh-keygen -t rsa -P ""
cat $HOME/.ssh/id_rsa.pub >>$HOME/.ssh/authorized_keys
```

Figura 25: Comandos configuración SSH en Linux

4.2.2.2. Configuración pseudo-distribuida

En esta configuración, toda la instalación se realiza en una sola máquina, por lo que la configuración de maestro y esclavo se realiza en el mismo lugar, que finalmente compartirán espacio dentro de la misma máquina, creando una separación maestro-esclavo virtual.

Los pasos a seguir para instalar Hadoop pseudo-distribuido [22] son los siguientes:

4.2.2.2.1. Descarga y preparación de la última versión estable

Accedemos a la página web de Apache para obtener la última versión estable de Hadoop [25]. La versión utilizada a lo largo de este documento es la 2.7.2. [Enero 2016].

Entre los varios formatos disponibles para descarga utilizamos el formato binario, identificado con la extensión de fichero "tar.gz", que nos proporciona una transmisión del fichero desde el servidor de Apache libre de errores.

Por tanto, el fichero descargado es el siguiente: hadoop-2.7.2.tar.gz

Una vez descargado, movemos el fichero a la ruta utilizada por convenio en este tipo de instalaciones: /usr/local/

Extraemos el fichero comprimido en la ruta indicada anteriormente y la renombramos a Hadoop asignándole al usuario hduser creado anteriormente los permisos de la nueva carpeta:

```
sudo tar -xzvf hadoop-2.7.2.tar.gz
sudo mv hadoop-2.7.2 /usr/local/hadoop
sudo chown hduser:hadoop -R /usr/local/hadoop
```

Figura 26: Comandos instalación Hadoop en Linux

Por tanto, la ruta final del sistema Hadoop una vez instalado es: /usr/local/hadoop

4.2.2.2.1. Creación de carpetas para archivos temporales

Dentro de la instalación de Hadoop es necesario crear una serie de carpetas para los archivos temporales generados por Hadoop una vez entre en funcionamiento. Para ello, creamos las siguientes carpetas y asignamos los permisos correspondientes al usuario hduser:

-Namenode: Carpeta que contendrá los archivos temporales creados por el proceso Namenode

-Datanode: Carpeta que contendrá los archivos temporales creados por el proceso Datanode

```
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

Figura 27: Creación carpetas temporales HDFS configuración pseudo-distribuida

4.2.2.2.1. Ficheros de configuración Hadoop

Una vez instalado Hadoop, procedemos a actualizar los ficheros de configuración para aplicar la configuración deseada, en este caso, la pseudo-distribuida.

-Fichero de configuración del usuario hduser: $ sudo gedit .bashrc

Incluimos la ruta dónde hemos instalado Java y Hadoop, así como las rutas de los diferentes elementos que componen el sistema:

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
```

Figura 28: Fichero de configuración del usuario hduser .bashrc

Todos los ficheros de configuración que vamos a modificar se encuentran en la ruta /usr/local/hadoop/etc/hadoop a la que accedemos mediante:

$ cd /usr/local/hadoop/etc/Hadoop

- Fichero de configuración del sistema Hadoop: core-site.xml

$ sudo gedit core-site.xml

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
    </property>
</configuration>
```

Figura 29: Fichero de configuración core-site.xml configuración pseudo-distribuida

-Fichero de configuración del sistema HDFS: hdfs-site.xml:

$ sudo gedit hdfs-site.xml

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
</configuration>
<property>
      <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
 </property>
 <property>
      <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
 </property>
```

Figura 30: Fichero de configuración hdfs-site.xml configuración pseudo-distribuida

-Fichero de configuración de las variables de entorno de Hadoop: hadoop-env.sh

$ sudo gedit hadoop-env.sh

```
JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Figura 31: Fichero de configuración hadoop-env.sh configuración pseudo-distribuida

-Fichero de configuración del sistema YARN: yarn-site.xml

$ sudo gedit yarn-site.xml

```
<property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
</property>
<property>
      <name>yarn.nodemanager.aux-
services.mapreduce.shuffle.class</name>
     <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Figura 32: Fichero de configuración yarn-site.xml configuración pseudo-distribuida

-Fichero de configuración de Mapreduce: mapred-site.xml

El fichero mapred-site.xml no existe como tal, sino que disponemos de una plantilla que puede ser utilizada o no, para hacer uso de ella, realizamos una copia de la plantilla y realizamos las modificaciones sobre la copia:

cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/ local/ hadoop/ etc/ hadoop/mapred-site.xml

$ sudo gedit mapred-site.xml

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
</configuration>
```

Figura 33: Fichero de configuración mapred-site.xml configuración pseudo-distribuida

Una vez realizada la configuración de todos los ficheros necesarios es necesario formatear el namenode antes de empezar a utilizar Hadoop:

$ bin/hdfs namenode -format

4.2.2.3. Configuración multinodo

A continuación se exponen los pasos de instalación para una configuración multinodo, en dónde el maestro se encuentra en una máquina y cada esclavo en otra diferente, habiendo N+1 máquinas, dónde N es el número de esclavos que queremos en nuestra infraestructura.

Para realizar esta configuración partimos de la configuración realizada en los apartados 4.2.2.1 y 4.2.2.2 y de una red de esclavos preparada con la misma instalación de Linux realizada en el apartado 4.2.1.

4.2.2.3.1. Configuración de las máquinas

-Asignación de direcciones IP y nombres de máquina en cluster doméstico:

Debemos asignar las direcciones IP y los nombres a cada una de las máquinas que deseamos que conformen nuestra red, para ello, añadimos los nombres de todos los

nodos en /etc/hosts en todas las máquinas que vayamos a emplear, tanto maestros como esclavos[2]:

$ sudo gedit /etc/hosts

```
127.0.0.1localhost
127.0.1.1aller
192.168.2.14    HadoopMaster
192.168.2.15    HadoopSlave1
192.168.2.16    HadoopSlave2
# The following lines are desirable for IPv6 capable
hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Figura 34: Fichero de configuración hosts

Además de editar el fichero anterior es necesario configurar las direcciones IP manualmente, para ello, nos dirigimos al apartado configuraciones de red en Linux en cada una de las máquinas de la siguiente manera:

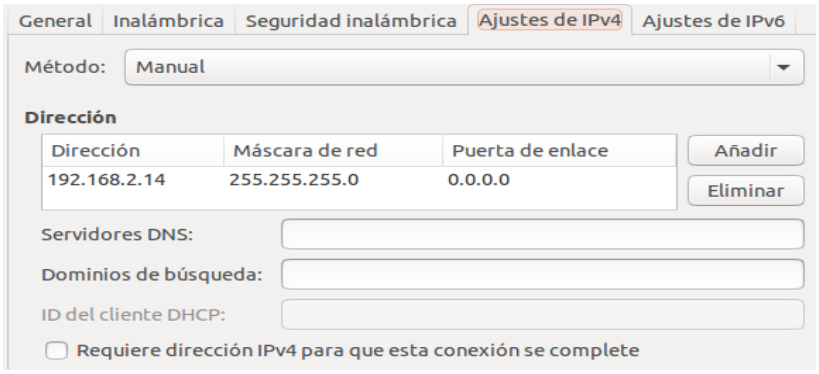


Figura 35: Configuración IP local en Linux para cluster doméstico

-Requisitos previos máquinas esclavo:

Los requisitos previos en las máquinas esclavo en esta configuración son los mismos que en la configuración pseudo-distribuida, por ello, se siguen los mismos pasos que en el apartado 4.2.2.1

[2] La configuración mostrada corresponde a una red de dos esclavos, para aumentar el número de esclavos se deben añadir las líneas correspondientes al fichero etc/hosts en cada máquina

-Configuración de privilegios: Añadimos ahora los permisos necesarios para ejecutar todas las operaciones bajo el usuario hduser:

```
# User privilege specification
rootALL=(ALL:ALL) ALL
# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
# Allow members of group sudo to execute any command
%sudoALL=(ALL:ALL) ALL
# Allow hduser of group hadoop to execute any command
hduser    ALL=(ALL:ALL) ALL
```

Figura 36: Configuración permisos de usuario en Linux

-Nombre del maestro:

Para tener un nombre más legible y facilitar tareas durante la utilización de Hadoop, cambiamos el nombre del hostname en /etc/hostname:

$ sudo gedit /etc/hostname

```
HadoopMaster
```

Figura 37: Edición nombre de localhost

-Aplicar los cambios:

Una vez finalizados los pasos anteriores es necesario reiniciar los equipos:

$ sudo reboot

4.2.2.3.2. Ficheros de configuración de Hadoop

Una vez instalado Hadoop, procedemos a actualizar los ficheros de configuración para aplicar la configuración deseada, en este caso, la multinodo.

Todos los ficheros de configuración que vamos a modificar se encuentran en la ruta /usr/local/hadoop/etc/hadoop a la que accedemos mediante:

$ cd /usr/local/hadoop/etc/Hadoop

- Fichero de configuración del sistema Hadoop: core-site.xml

$ sudo gedit core-site.xml

```
<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/usr/local/hadoop/nuevotmp</value>
<description>A base for other temporary
directories.</description>
</property>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://HadoopMaster:9000</value>
    </property>
</configuration>
```

Figura 38: Fichero de configuración core-site.xml configuración multinodo

$ sudo gedit hdfs-site.xml

```
<configuration>
<property>
   <name>dfs.replication</name>
   <value>3</value>
 </property>
 <property>
   <name>dfs.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Figura 39: Fichero de configuración hdfs-site.xml configuración multinodo

- Fichero de configuración del sistema YARN: yarn-site.xml

$ sudo gedit yarn-site.xml

```
<configuration>
<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>HadoopMaster:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>HadoopMaster:8035</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>HadoopMaster:8050</value>
</property>
</configuration>
```

Figura 40: Fichero de configuración yarn-site.xml configuración multinodo

$ sudo gedit mapred-site.xml

```
<configuration>
<property>
<name>mapreduce.job.tracker</name>
<value>HadoopMaster:5431</value>
</property>
 <property>
   <name>mapreduce.framework.name</name>
   <value>yarn</value>
</property>
</configuration>
```

Figura 41: Fichero de configuración mapred-site.xml configuración multinodo

- Fichero de configuración de Maestros: master

$ sudo gedit master

```
## Add name of master nodes
HadoopMaster
```

Figura 42: Fichero de configuración maestro configuración multinodo

- Fichero de configuración de Esclavos: slaves

$ sudo gedit slaves

```
## Add name of slave nodes
HadoopSlave1
HadoopSlave2
```

Figura 43: Fichero de configuración esclavos configuración multinodo

4.2.2.3.3. Replicación del sistema de archivos y carpetas

Una vez realizada la configuración deseada en el maestro, es necesaria copiar todas las carpetas y archivos de la instalación de Hadoop en todos los esclavos. Para ello utilizamos rsync, instalado anteriormente.

La instalación de Hadoop se encuentra en una sola carpeta en la ruta /usr/local/hadoop, por lo que el contenido que encontramos en ella es el que hay que replicar en todos los esclavos. Para ello, utilizamos la siguiente instrucción desde el maestro:

```
$ sudo rsync -avxP /usr/local/hadoop/ hduser@HadoopSlave1:/usr/local/hadoop/
$ sudo rsync -avxP /usr/local/hadoop/ hduser@HadoopSlave2:/usr/local/hadoop/
```

Figura 44: Copia de archivos Hadoop desde el maestro hacia los esclavos

En el caso de haber más esclavos, sería necesario aplicar la instrucción anterior por cada uno de ellos.

4.2.2.3.4. Modificación de carpetas para archivos temporales

En la configuración pseudo-distribuida creamos dos carpetas temporales; namenode y datanode, sin embargo, en la configuración multinodo el datanode se encuentra en los esclavos y por tanto debemos eliminarla del maestro:

-Maestro:

```
sudo rm -rf /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

Figura 45: Creación carpetas para archivos temporales HDFS multinodo en maestro

-Esclavos:

```
sudo rm -rf /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
sudo chown hduser:hadoop -R /usr/local/hadoop_tmp/
```

Figura 46: Creación carpetas para archivos temporales HDFS multinodo en los esclavos

4.2.2.3.5. Configuración SSH

Tenemos que realizar la configuración ssh con el objetivo de compartir los certificados que se crean y así poder establecer las comunicaciones entre todos los nodos.

Comandos a ejecutar en la consola de comandos de Linux para la configuración ssh:

```
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave1
$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@HadoopSlave2
```

Figura 47: Distribución claves ssh entre máquinas

Una vez realizada la configuración de todos los ficheros necesarios es necesario formatear el namenode en el maestro antes de empezar a utilizar Hadoop:

$ bin/hdfs namenode -format

4.2.3. Instalación Hive

La instalación de Hive se realiza de una forma menos tediosa que Hadoop por dos motivos principales; porque es un complemento de Hadoop que funciona directamente encima de él y porque sólo es necesario instalarlo en el maestro para poder funcionar. Los pasos a seguir para la instalación son los que se exponen en los subapartados siguientes.

4.2.3.1. Prerrequisitos

Los requisitos previos [26] para poder instalar Hive son los siguientes:

- Tener instalado al menos Java 1.7
- Tener instalado Hadoop. Preferiblemente a partir de las versiones 2.x

4.2.3.2. Descarga y preparación de la última versión estable

Accedemos a la página web de Apache para obtener la última versión estable de Hive [27]. La versión utilizada a lo largo de este documento es la 1.2.1. [Junio 2015].

Entre los varios formatos disponibles para descarga utilizamos el formato binario, identificado con la extensión de fichero “tar.gz”, que nos proporciona una transmisión del fichero desde el servidor de Apache libre de errores.

Por tanto, el fichero descargado es el siguiente: apache-hive-1.2.1-bin.tar.gz

Una vez descargado, movemos el fichero a la ruta utilizada por convenio en este tipo de instalaciones: /usr/local/hive

Extraemos el fichero comprimido en la ruta indicada anteriormente y la renombramos a Hadoop asignándole al usuario hduser creado anteriormente los permisos de la nueva carpeta:

```
sudo tar -xzvf apache-hive-1.2.1-bin.tar.gz
sudo mv apache-hive-1.2.1-bin /usr/local/hive
sudo chown hduser:hadoop -R /usr/local/hive
```

Figura 48: Instalación Hive

Por tanto, la ruta final una vez instalado Hive es: /usr/local/hive

4.2.3.3. Configuración de las variables de entorno

Una vez instalado Hive procedemos a actualizar las variables de entorno del sistema para indicar la ruta de nuestra instalación y así poder ejecutar Hive desde cualquier carpeta del sistema: $ sudo gedit .bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export HIVE_HOME=/usr/local/hadoop/hive
export PATH=$PATH:$HIVE_HOME/bin
```

Figura 49: Configuración de las variables de entorno para Hive

4.2.3.3.4. Creación de carpetas para archivos Hive

Para poder crear tablas en Hive es requisito indispensable la creación de una serie de carpetas para diferentes tipos de archivos, tanto temporales como archivos propios de las tablas en HDFS. Para realizar dicha acción seguimos las siguientes instrucciones:

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir       /tmp
$ $HADOOP_HOME/bin/hadoop fs -mkdir       /user/hive/warehouse
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w   /tmp
$ $HADOOP_HOME/bin/hadoop fs -chmod g+w   /user/hive/warehouse
```

Figura 50: Creación de carpetas Hive en HDFS

4.3. Obtención de los datos

4.3.1. Introducción

Una vez instalada toda la infraestructura Hadoop-Hive el objetivo es la obtención de los datos con los que vamos a trabajar, en este caso con datos provenientes de los mercados financieros más relevantes; Nasdaq, Nikkei y Ftse y además con el mercado financiero nacional, Ibex 35. Para poder trabajar de una manera más eficiente cada mercado financiero será almacenado en un fichero diferente, trataremos cada mercado por separado en primera instancia y los combinaremos más tarde para medir la eficiencia del sistema al juntar varios archivos.

4.3.2. Fuentes de información

El objetivo a la hora de buscar los datos era el de encontrar una fuente fiable y continuamente actualizada para poder analizar datos muy recientes. Además de dichos datos recientes se precisaba un histórico de datos de los diferentes mercados financieros para poder llevar a cabo nuestro análisis de inversión. Tras analizar varias alternativas, algunas de ellas descartadas por proveer mecanismos de descarga con datos incompletos y otras por datos erróneos, el sitio web del que obtenemos toda la información es Yahoo Finance [1].

A continuación se muestra una tabla con los diferentes ficheros creados a partir de las url de Yahoo Finance, dónde se detalla el número de datos (número de filas) del fichero, el periodo de tiempo que comprenden esos datos, el número de valores que forman dicho mercado financiero, el tamaño del fichero una vez ha sido descargado y por último, el nombre del fichero dónde se almacena.

Mercado	Nº datos	Años	Nº valores	Tamaño fichero (kB)	Nombre fichero	Fuente (URL)
NASDAQ Composite (^IXIC)	11388	05/02/71-2016	2588	840,7	nasdaq.csv	http://real-chart.finance.yahoo.com/table.csv?s=%5EIXIC
FTSE 100 (^FTSE)	7504	03/01/84-2016	101	625,5	ftse.csv	http://real-chart.finance.yahoo.com/table.csv?s=%5EFTSE
Nikkei 225 (^N225)	7949	04/01/84-2016	225	565	nikkei.csv	http://real-chart.finance.yahoo.com/table.csv?s=%5EN225

IBEX 35 (^IBEX)	5813	15/02/93-2016	35	439,5	Ibex.csv	http://real-chart.finance.yahoo.com/table.csv?s=%5EIBEX

Tabla 2: Fuentes y detalles de la información empleada

4.3.3. Procedimiento

El procedimiento de obtención de los datos se realiza siguiendo la url proporcionada por Yahoo Finance.

-Descarga de datos hasta el día actual:

Para descargar todos los datos disponibles hasta el día actual tan sólo hace falta ir hasta la siguiente url, añadiendo tan sólo la terminación del mercado en el que estemos interesados:

http://real-chart.finance.yahoo.com/table.csv?s=%5EIXIC

-Descarga de datos de un intervalo de tiempo seleccionado:

Para descargar los datos disponibles en un intervalo de tiempo determinado es necesario indicarlo en la url.

Cómo ejemplo tenemos la siguiente dirección:

http://real-chart.finance.yahoo.com/table.csv?s=%5EIXIC&a=01&b=5&c=1971&d=01&e=6&f=2016&g=d&ignore=.csv

En ella se ha añadido:

- Fecha de inicio: 2/5/1971. Con el formato: &a=01&b=5&c=1971
- Fecha de fin: 2/6/2016. Con el formato: &d=01&e=6&f=2016

Todos los ficheros descargados desde Yahoo Finance vienen en formato csv, formato admitido por Hive, por tanto no es necesario realizar ninguna conversión para introducir los datos en nuestro sistema Hadoop.

4.4. Transformación y almacenamiento de los datos

4.4.1. Introducción

Una vez han sido descargados los datos, tal y como explicamos en el apartado 4.3, procedemos a visualizar la estructura del fichero descargado para así poder introducir los datos en el sistema.

Cabecera	Descripción
Date	Día al que corresponden los datos de una fila
Open	Valor de apertura del mercado
High	Valor más alto del mercado
Low	Valor más abajo del mercado
Close	Valor de cierre del mercado
Volume	Volumen de operaciones sobre el mercado
Adj Close	Ajuste del valor del mercado posterior al cierre de la sesión

Tabla 3: Estructura archivo csv mercados financieros Yahoo Finance

A continuación se muestran los datos correspondientes a una jornada del mercado con sus correspondientes cabeceras

Date,Open,High,Low,Close,Volume,Adj Close
2016-06-10,4915.149902,4917.919922,4880.609863,4894.549805,1822200000,4894.549805

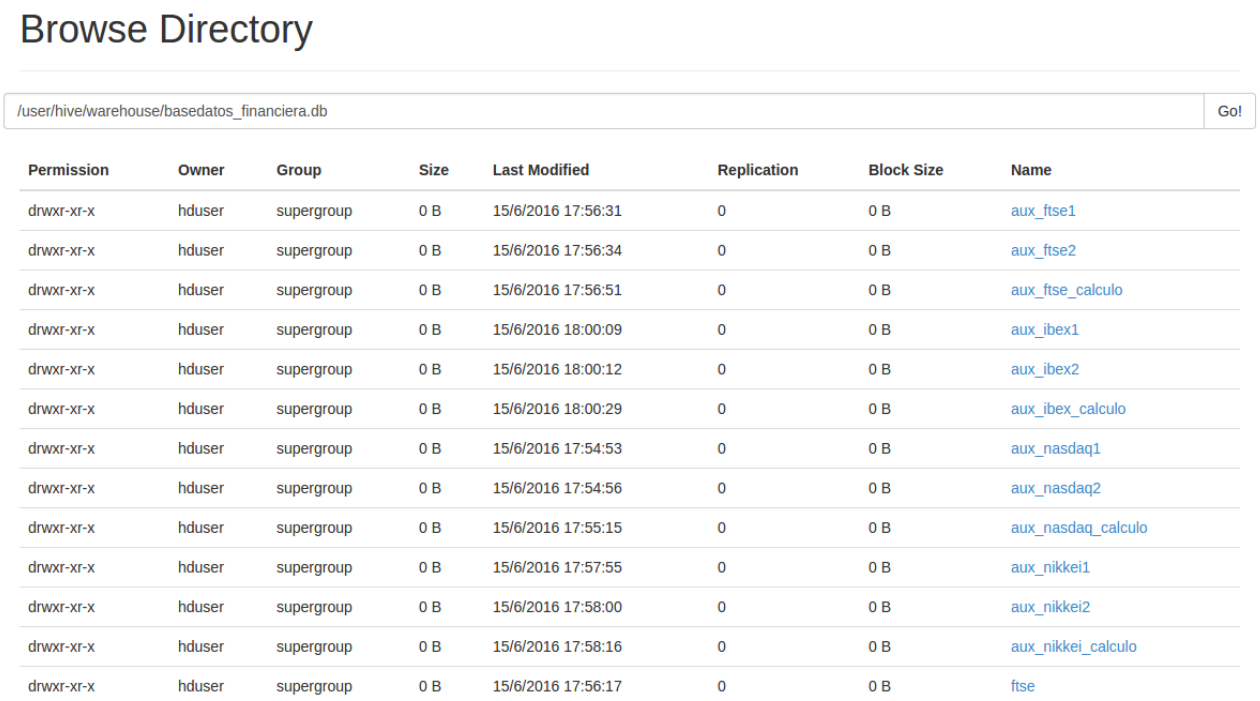
Figura 51: Ejemplo de datos procedente de un archivos csv de mercados financieros

Toda la gestión de ficheros y carpetas se realiza gracias a Hive, que actúa como aplicación externa encima de Hadoop, permitiendo realizar operaciones y consultas sobre el sistema de archivos de HDFS. Por tanto, nuestro almacén de datos es gestionado íntegramente por Hive.

Todos los datos creados e importados, así como las operaciones de creación de tablas se almacenan en una carpeta denominada mestastore, que se genera de manera automática en el lugar desde dónde se está ejecutando Hive. Por tanto, para poder acceder a todos nuestros datos es muy importante ejecutar Hive desde el mismo sitio cada vez que arranquemos el sistema, si no, todos los datos generados no estarán disponibles y se creará una nueva carpeta metastore.

4.4.2. Procedimiento

Para llevar un orden en el análisis de datos, creamos un sistema de carpetas en nuestro almacén de datos a través de HiveQL, como se muestra a continuación:

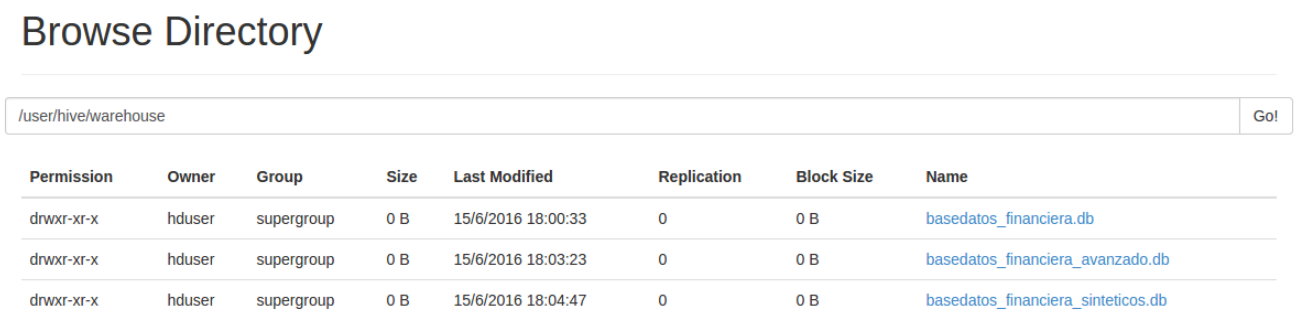


Browse Directory

/user/hive/warehouse/basedatos_financiera.db Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:56:31	0	0 B	aux_ftse1
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:56:34	0	0 B	aux_ftse2
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:56:51	0	0 B	aux_ftse_calculo
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:00:09	0	0 B	aux_ibex1
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:00:12	0	0 B	aux_ibex2
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:00:29	0	0 B	aux_ibex_calculo
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:54:53	0	0 B	aux_nasdaq1
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:54:56	0	0 B	aux_nasdaq2
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:55:15	0	0 B	aux_nasdaq_calculo
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:57:55	0	0 B	aux_nikkei1
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:58:00	0	0 B	aux_nikkei2
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:58:16	0	0 B	aux_nikkei_calculo
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 17:56:17	0	0 B	ftse

Figura 53: Estructura de tablas en Hive para almacenamiento de datos



Browse Directory

/user/hive/warehouse Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:00:33	0	0 B	basedatos_financiera.db
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:03:23	0	0 B	basedatos_financiera_avanzado.db
drwxr-xr-x	hduser	supergroup	0 B	15/6/2016 18:04:47	0	0 B	basedatos_financiera_sinteticos.db

Figura 52: Estructura de bases de datos en Hive para el almacenamiento de tablas

Para poder crear las bases de datos indicadas, ejecutamos el siguiente comando una vez hemos arrancado la consola Hive:

```
create database IF NOT EXISTS nombre_base_datos;
```

Figura 54: Creación de bases de datos en Hive

Para crear las tablas que contendrán los datos de los diferentes mercados financieros, ejecutamos los siguientes comandos una vez hemos arrancado la consola Hive:

```
use nombre_base_datos;
create table IF NOT EXISTS nombre_tabla (fecha string,open
float,high float,low float,close float,volume float,adjclose
float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
load data local inpath
'/home/miguelaller/Documentos/TFG/Datos/nombre_fichero.csv'
overwrite into table nombre_tabla;
```

Figura 55: Creación de tablas e importación de datos en Hive

Como vemos, para poder crear una tabla es necesario seguir una serie de pasos:

-Indicar la base de datos en dónde quieres crear la tabla

-Crear la tabla indicando el nombre y el tipo de los campos, que tiene que corresponder con los del fichero descargado de Finance Yahoo, cuya estructura se analizó en la Figura 51.

-Los ficheros CSV delimitan sus campos por el símbolo ";" una vez se tratan como texto plano, como ocurre en Hive, por ello se indica en la instrucción el tipo de limitador que contiene el fichero.

-Cargar los datos desde el fichero deseado, el cual se encuentra almacenado en nuestra máquina maestro, hacia una de las tablas creadas anteriormente.

4.4. Análisis y tratado de datos mediante scripts HiveQL

Una vez los datos han sido cargados en el sistema Hadoop a través de Hive, podemos tratarlos con instrucciones HiveQL en la línea de comandos de la consola de Hive.

Para automatizar los procesos de tratado de datos y evitar así escribir las instrucciones una a una en la consola de comandos se utilizan los denominados script HQL [28], que son ficheros que contienen instrucciones en lenguaje HiveQL en cada una de sus líneas, las cuales se ejecutan de manera secuencial una vez se ha ejecutado el archivo.

Para el tratado de datos se han creado tres algoritmos basados en instrucciones HiveQL, cada uno de los cuáles está basado en directrices de inversión financiera, con la finalidad de obtener la máxima rentabilidad monetaria en cada uno de ellos a partir de los datos provenientes de mercados financieros. En los subapartados siguientes se exponen los algoritmos generados.

4.5.1. Algoritmo básico de estrategia de inversión

Para la implementación de la estrategia de inversión básica vamos a emplear la siguiente regla: Si el valor del mercado financiero sube al finalizar una jornada, la tendencia que

seguirá la jornada siguiente será también alcista y por tanto invertiremos, si por el contrario el valor del mercado financiero baja o se mantiene al finalizar una jornada, la tendencia que seguirá la jornada siguiente será bajista y por tanto no invertiremos. Dicha regla procede de una simplificación del análisis técnico bursátil y de la teoría de las tendencias económicas en los mercados financieros [29] [30].

En la siguiente figura, se muestra a modo de esquema la estrategia de inversión seguida en el algoritmo de inversión básico:

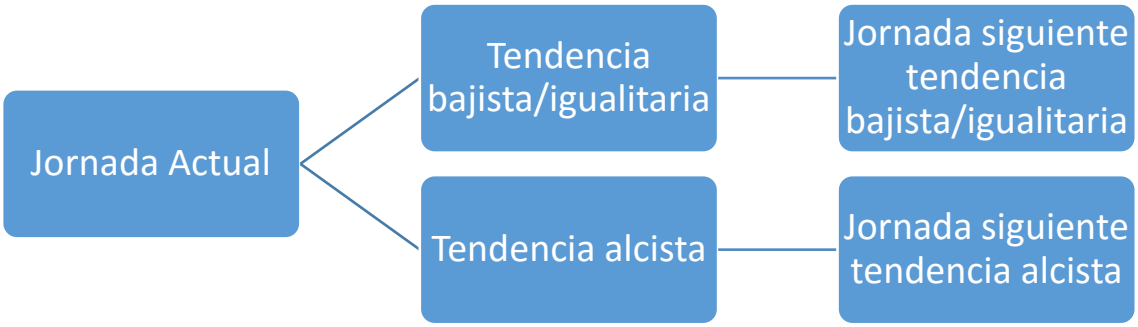


Figura 56: Estrategia de inversión algoritmo básico

Con nuestra base de datos ya creada, accedemos a ella, guardamos los datos del Nasdaq, Ibex, Ftse y Nikkei hasta el día actual y procedemos con el algoritmo que nos dará como resultado el porcentaje de acierto del mismo teniendo en cuenta los datos históricos.

Los pasos seguidos en los algoritmos mostrados en la Tabla 5 son los siguientes:

1. Creación de la base de datos en caso de no existir
2. Creación de la tabla a utilizar con los campos correspondientes (véase apartado 4.4.2) en caso de no existir
3. Introducción de los datos provenientes del mercado financiero escogido en la tabla creada anteriormente
4. Creación de dos tablas auxiliares para almacenar en una de ellas los valores originales y en la otra los valores desfasados un día para cumplir con nuestra estrategia de inversión.

   En este punto, los datos originales ya han sido filtrados, y de la tabla de datos original quedan los datos que no se encuentran tachados:

Cabecera	Descripción
Date	Día al que corresponden los datos de una fila
Open	Valor de apertura del mercado
High	Valor más alto del mercado
Low	Valor más abajo del mercado
Close	Valor de cierre del mercado
Volume	Volumen de operaciones sobre el mercado
Adj Close	Ajuste del valor del mercado posterior al cierre de la sesión

Tabla 4: Filtrado datos originales

5. Unificación de las dos tablas auxiliares mediante el comando HiveQL join
6. Creación de una tabla auxiliar para obtener el acierto de cada jornada por separado
7. Por último se realiza una contabilización de los aciertos y una simple operación para mostrar los resultados en forma de porcentaje.

En la Tabla 5 se puede ver la relación entre el mercado financiero y su correspondiente fichero que contiene el algoritmo empleado:

Mercado	Archivo (Script Hql)
Nasdaq	algoritmo_base_nasdaq.hql
Ftse	algoritmo_base_ftse.hql
Nikkei	algoritmo_base_nikkei.hql
Ibex 35	algoritmo_base_ibex35.hql

Tabla 5: Relación mercados financieros - scripts creados

La razón por la cual existe un algoritmo diferente para cada mercado es la de obtener el mayor ahorro computacional posible. Las únicas diferencias existentes entre unos algoritmos y otros son:

-Fecha de inicio y fin de los datos.

-Nombres de los archivos desde donde se cargan los datos.

-Nombres de las tablas empleadas.

Dichas diferencias se pueden observar en el apartado siguiente (4.5.1.1)

4.5.1.1. Implementación de los diferentes algoritmos básicos de inversión

algoritmo_base_nasdaq.hql

```
create database IF NOT EXISTS basedatos_financiera;
use basedatos_financiera;
create table IF NOT EXISTS nasdaq (fecha string,open float,high float,low
float,close float,volume float,adjclose float) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

load data local inpath '/home/miguelaller/Documentos/TFG/Datos/nasdaq.csv'
overwrite into table nasdaq;

drop table IF EXISTS aux_nasdaq1;
create table IF NOT EXISTS aux_nasdaq1 (id int,dif float);
drop table IF EXISTS aux_nasdaq2;
create table IF NOT EXISTS aux_nasdaq2 (id int,predic float);

insert into aux_nasdaq1 (id,dif) select row_number() over(),close-open from
nasdaq where nasdaq.fecha >= '1971-02-08' and  nasdaq.fecha <='2016-04-01';

insert into aux_nasdaq2 (id,predic) select row_number() over(),close-open from
nasdaq where nasdaq.fecha >= '1971-02-05' and  nasdaq.fecha <='2016-03-31';

drop table IF EXISTS nasdaq_calculo;
create table IF NOT EXISTS nasdaq_calculo AS
select n1.dif,n2.predic
from aux_nasdaq1 n1
inner join aux_nasdaq2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_nasdaq_calculo;
create table IF NOT EXISTS aux_nasdaq_calculo AS
select dif*predic as aux_resultado
from nasdaq_calculo;

drop table IF EXISTS resultado_nasdaq;
create table IF NOT EXISTS resultado_nasdaq (resultado float);
insert into resultado_nasdaq (resultado) select count(*) from
aux_nasdaq_calculo where aux_resultado>=0;

select (resultado/11387)*100 from resultado_nasdaq;
```

Figura 57: Algoritmo de inversión básico Nasdaq

Para poder ejecutar el script creado y explicado anteriormente tenemos que utilizar la siguiente sentencia en la consola de comandos:

hive -f algoritmo_base_nasdaq.hql

algoritmo_base_ftse.hql

```
create database IF NOT EXISTS basedatos_financiera;
use basedatos_financiera;
create table IF NOT EXISTS ftse (fecha string,open float,high float,low
float,close float,volume float,adjclose float) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

load data local inpath '/home/miguelaller/Documentos/TFG/Datos/ftse.csv'
overwrite into table ftse;

drop table IF EXISTS aux_ftse1;
create table IF NOT EXISTS aux_ftse1 (id int,dif float);
drop table IF EXISTS aux_ftse2;
create table IF NOT EXISTS aux_ftse2 (id int,predic float);

insert into aux_ftse1 (id,dif) select row_number() over(),close-open from
ftse where ftse.fecha >= '1984-01-04' and  ftse.fecha <='2016-04-19';

insert into aux_ftse2 (id,predic) select row_number() over(),close-open from
ftse where ftse.fecha >= '1984-01-03' and  ftse.fecha <='2016-04-18';

drop table IF EXISTS ftse_calculo;
create table IF NOT EXISTS ftse_calculo AS
select n1.dif,n2.predic
from aux_ftse1 n1
inner join aux_ftse2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_ftse_calculo;
create table IF NOT EXISTS aux_ftse_calculo AS
select dif*predic as aux_resultado
from ftse_calculo;

drop table IF EXISTS resultado_ftse;
create table IF NOT EXISTS resultado_ftse (resultado float);
insert into resultado_ftse (resultado) select count(*) from aux_ftse_calculo
where aux_resultado>=0;

select (resultado/7503)*100 from resultado_ftse;
```

Figura 58: Algoritmo de inversión básico Ftse

Para poder ejecutar el script creado y explicado anteriormente tenemos que utilizar la siguiente sentencia en la consola de comandos:

hive -f algoritmo_base_ftse.hql

algoritmo_base_nikkei.hql

```
create database IF NOT EXISTS basedatos_financiera;
use basedatos_financiera;
create table IF NOT EXISTS nikkei (fecha string,open float,high float,low
float,close float,volume float,adjclose float) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

load data local inpath '/home/miguelaller/Documentos/TFG/Datos/nikkei.csv'
overwrite into table nikkei;

drop table IF EXISTS aux_nikkei1;
create table IF NOT EXISTS aux_nikkei1 (id int,dif float);
drop table IF EXISTS aux_nikkei2;
create table IF NOT EXISTS aux_nikkei2 (id int,predic float);

insert into aux_nikkei1 (id,dif) select row_number() over(),close-open from
nikkei where nikkei.fecha >= '1984-01-05' and  nikkei.fecha <='2016-04-12';

insert into aux_nikkei2 (id,predic) select row_number() over(),close-open
from nikkei where nikkei.fecha >= '1984-01-04' and  nikkei.fecha <='2016-04-
11';

drop table IF EXISTS nikkei_calculo;
create table IF NOT EXISTS nikkei_calculo AS
select n1.dif,n2.predic
from aux_nikkei1 n1
inner join aux_nikkei2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_nikkei_calculo;
create table IF NOT EXISTS aux_nikkei_calculo AS
select dif*predic as aux_resultado
from nikkei_calculo;

drop table IF EXISTS resultado_nikkei;
create table IF NOT EXISTS resultado_nikkei (resultado float);
insert into resultado_nikkei (resultado) select count(*) from
aux_nikkei_calculo where aux_resultado>=0;

select (resultado/7948)*100 from resultado_nikkei;
```

Figura 59: Algoritmo de inversión básico Nikkei

Para poder ejecutar el script creado y explicado anteriormente tenemos que utilizar la siguiente sentencia en la consola de comandos:

hive -f algoritmo_base_nikkei.hql

algoritmo_base_ibex.hql

```
create database IF NOT EXISTS basedatos_financiera;
use basedatos_financiera;
create table IF NOT EXISTS ibex (fecha string,open float,high float,low
float,close float,volume float,adjclose float) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';

load data local inpath '/home/miguelaller/Documentos/TFG/Datos/ibex.csv'
overwrite into table ibex;

drop table IF EXISTS aux_ibex1;
create table IF NOT EXISTS aux_ibex1 (id int,dif float);
drop table IF EXISTS aux_ibex2;
create table IF NOT EXISTS aux_ibex2 (id int,predic float);

insert into aux_ibex1 (id,dif) select row_number() over(),close-open from
ibex where ibex.fecha >= '1993-03-19' and  ibex.fecha <='2016-04-13';

insert into aux_ibex2 (id,predic) select row_number() over(),close-open from
ibex where ibex.fecha >= '1993-02-15' and  ibex.fecha <='2016-04-12';

drop table IF EXISTS ibex_calculo;
create table IF NOT EXISTS ibex_calculo AS
select n1.dif,n2.predic
from aux_ibex1 n1
inner join aux_ibex2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_ibex_calculo;
create table IF NOT EXISTS aux_ibex_calculo AS
select dif*predic as aux_resultado
from ibex_calculo;

drop table IF EXISTS resultado_ibex;
create table IF NOT EXISTS resultado_ibex (resultado float);
insert into resultado_ibex (resultado) select count(*) from aux_ibex_calculo
where aux_resultado>=0;

select (resultado/5812)*100 from resultado_ibex;
```

Figura 60: Algoritmo de inversión básico Ibex

Para poder ejecutar el script creado y explicado anteriormente tenemos que utilizar la siguiente sentencia en la consola de comandos:

hive -f algoritmo_base_ibex.hql

4.5.1.2. Simplificación del algoritmo

Para no depender de un algoritmo por cada mercado financiero, hemos realizado una simplificación y generalización del mismo par así depender de un solo algoritmo que nos sirve para todos los mercados.

El cambio principal entre los algoritmos anteriores y el actual simplificado se produce a la hora de crear las tablas auxiliares e introducir los datos en ellas. Anteriormente se filtraban los datos deseados por fecha, lo que limitaba el uso generalizado, por lo que en este caso los datos deseados se filtran por id, proporcionándonos así el algoritmo versátil que buscábamos: **algoritmo_base_simplificado.hql**

```
set longitud=(longitud del fichero a analizar);

create database IF NOT EXISTS basedatos_financiera;
use basedatos_financiera;

create table IF NOT EXISTS mercado (fecha string,open float,high float,low
float,close float,volume float,adjclose float) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/(mercado a
analizar).csv' overwrite into table mercado;

drop table IF EXISTS aux_mercado;
create table IF NOT EXISTS aux_mercado (id int,dif float);
insert into aux_mercado (id,dif) select row_number() over(),close-open from
mercado;

drop table IF EXISTS aux_mercado1;
create table IF NOT EXISTS aux_mercado1 (id int,dif float);
drop table IF EXISTS aux_mercado2;
create table IF NOT EXISTS aux_mercado2 (id int,predic float);

insert into aux_mercado1 (id,dif) select row_number() over(),dif from
aux_mercado where aux_mercado.id >=1   and  aux_mercado.id
<='${hiveconf:longitud}'-1;
insert into aux_mercado2 (id,predic) select row_number() over(),dif from
aux_mercado where aux_mercado.id >=2   and  aux_mercado.id
<='${hiveconf:longitud}';

drop table IF EXISTS mercado_calculo;
create table IF NOT EXISTS mercado_calculo AS
select n1.dif,n2.predic
from aux_mercado1 n1
inner join aux_mercado2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_mercado_calculo;
create table IF NOT EXISTS aux_mercado_calculo AS
select dif*predic as aux_resultado
from mercado_calculo;

drop table IF EXISTS resultado_mercado;
create table IF NOT EXISTS resultado_mercado (resultado float);
insert into resultado_mercado (resultado) select count(*) from aux_mercado_
calculo where aux_resultado>=0;

select (resultado/('${hiveconf:longitud}'-1))*100 from resultado_mercado;
```

Figura 61: Algoritmo de inversión básico simplificado

4.5.2. Algoritmo avanzado de estrategia de inversión

Al encontrarnos en un mundo globalizado, las tendencias económicas entre mercados están absolutamente ligadas, produciéndose movimientos en cadena y tendencias económicas similares entre las diferentes bolsas, afectadas entre sí por diferentes acontecimientos socio-económicos. Estos conceptos resumen el denominado "*efecto mariposa*" [31], cuyo significado queda reflejado perfectamente con la frase "*el aleteo de una mariposa se puede sentir al otro lado del mundo*". Por ello realizamos ahora una estrategia de inversión basada en estos conceptos.

Realizamos ahora un algoritmo de estrategia de inversión avanzada para ver cómo se afectan entre sí los diferentes mercados financieros a partir del cruce de tablas de datos.

La estrategia trazada consiste en ver la tendencia del Nasdaq y del Ibex del día anterior y la tendencia del Nikkei el día actual para ver cómo afectan en el Ibex. Esta lógica es debida a que el mercado japonés (Nikkei) cierra antes de que el español (Ibex 35) realice su apertura, por lo que los datos más recientes y definitivos del Nikkei son de la misma jornada, mientras que los del propio Ibex y los del Nasdaq son de la jornada anterior, quedando la relación entre mercados de la siguiente manera:

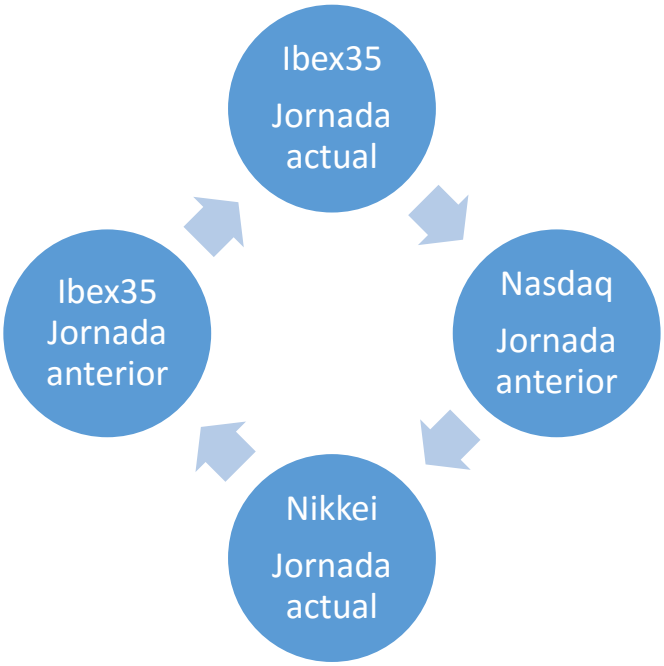


Figura 62: Estrategia de inversión avanzada

La estrategia presente en el algoritmo es por tanto la siguiente: Si alguno de los tres mercados empleados sigue una tendencia bajista o mantenida al finalizar una jornada, la tendencia que seguirá la jornada siguiente será bajista y por tanto no invertiremos.

Al igual que en el caso del algoritmo básico, en esta ocasión también emplearemos un script para facilitar el proceso de obtención y análisis de datos en Hive.

A continuación se expone el algoritmo presente en el archivo **algoritmo_avanzado.hql**:

```
set longitud=5813;
create database IF NOT EXISTS basedatos_financiera_avanzado;
use basedatos_financiera_avanzado;

create table IF NOT EXISTS mercado_avanzado_ibex (fecha string,open float,high
float,low float,close float,volume float,adjclose float) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/ibex.csv'
overwrite into table mercado_avanzado_ibex;

create table IF NOT EXISTS mercado_avanzado_nasdaq (fecha string,open
float,high float,low float,close float,volume float,adjclose float) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/nasdaq.csv'
overwrite into table mercado_avanzado_nasdaq;

create table IF NOT EXISTS mercado_avanzado_nikkei (fecha string,open
float,high float,low float,close float,volume float,adjclose float) ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/nikkei.csv'
overwrite into table mercado_avanzado_nikkei;

drop table IF EXISTS aux_mercado_avanzado_ibex;
create table IF NOT EXISTS aux_mercado_avanzado_ibex (id int,dif float);
insert into aux_mercado_avanzado_ibex (id,dif) select row_number()
over(),close-open from mercado_avanzado_ibex;

drop table IF EXISTS aux_mercado_avanzado_nasdaq;
create table IF NOT EXISTS aux_mercado_avanzado_nasdaq (id int,dif float);
insert into aux_mercado_avanzado_nasdaq (id,dif) select row_number()
over(),close-open from mercado_avanzado_nasdaq;

drop table IF EXISTS aux_mercado_avanzado_nikkei;
create table IF NOT EXISTS aux_mercado_avanzado_nikkei (id int,dif float);
insert into aux_mercado_avanzado_nikkei (id,dif) select row_number()
over(),close-open from mercado_avanzado_nikkei;

drop table IF EXISTS aux_mercado_avanzado_ibex1;
create table IF NOT EXISTS aux_mercado_avanzado_ibex1 (id int,dif float);
drop table IF EXISTS aux_mercado_avanzado_ibex2;
create table IF NOT EXISTS aux_mercado_avanzado_ibex2 (id int,predic float);

drop table IF EXISTS aux_mercado_avanzado_nasdaq1;
create table IF NOT EXISTS aux_mercado_avanzado_nasdaq1 (id int,predic float);
drop table IF EXISTS aux_mercado_avanzado_nikkei1;
create table IF NOT EXISTS aux_mercado_avanzado_nikkei1 (id int,dif float);
```

```
insert into aux_mercado_avanzado_ibex1 (id,dif) select row_number() over(),dif
from aux_mercado_avanzado_ibex where aux_mercado_avanzado_ibex.id >=1   and
aux_mercado_avanzado_ibex.id <='${hiveconf:longitud}'-1;

insert into aux_mercado_avanzado_ibex2 (id,predic) select row_number()
over(),dif from aux_mercado_avanzado_ibex where aux_mercado_avanzado_ibex.id
>=2   and  aux_mercado_avanzado_ibex.id <='${hiveconf:longitud}';

insert into aux_mercado_avanzado_nasdaq1 (id,predic) select row_number()
over(),dif from aux_mercado_avanzado_nasdaq where
aux_mercado_avanzado_nasdaq.id >=2   and  aux_mercado_avanzado_nasdaq.id
<='${hiveconf:longitud}';

insert into aux_mercado_avanzado_nikkei1 (id,dif) select row_number()
over(),dif from aux_mercado_avanzado_nikkei where
aux_mercado_avanzado_nikkei.id >=1   and  aux_mercado_avanzado_nikkei.id
<='${hiveconf:longitud}'-1;

drop table IF EXISTS aux_mercado_avanzado_ibex1_calc;
create table IF NOT EXISTS aux_mercado_avanzado_ibex1_calc (id int,calc3 int);
insert into aux_mercado_avanzado_ibex1_calc select id,
case when predic<0 then 0
else 1
End
from aux_mercado_avanzado_ibex2;

drop table IF EXISTS aux_mercado_avanzado_nikkei1_calc;
create table IF NOT EXISTS aux_mercado_avanzado_nikkei1_calc (id int,calc int);
insert into aux_mercado_avanzado_nikkei1_calc select id,
case when dif<0 then 0
else 1
End
from aux_mercado_avanzado_nikkei1;

drop table IF EXISTS aux_mercado_avanzado_nasdaq1_calc;
create table IF NOT EXISTS aux_mercado_avanzado_nasdaq1_calc (id int,calc2
int);
insert into aux_mercado_avanzado_nasdaq1_calc select id,
case when predic<0 then 0
else 1
End
from aux_mercado_avanzado_nasdaq1;

drop table IF EXISTS mercado_avanzado_calculo_aux;
create table IF NOT EXISTS mercado_avanzado_calculo_aux AS
select n1.id,n1.calc,n2.calc2
from aux_mercado_avanzado_nikkei1_calc n1
inner join aux_mercado_avanzado_nasdaq1_calc n2
on n1.id=n2.id;
```

```
drop table IF EXISTS mercado_avanzado_calculo_aux2;
create table IF NOT EXISTS mercado_avanzado_calculo_aux2 AS
select n1.id,n1.calc3,n2.calc,n2.calc2
from aux_mercado_avanzado_ibex1_calc n1
inner join mercado_avanzado_calculo_aux n2
on n1.id=n2.id;

drop table IF EXISTS mercado_avanzado_calculo_aux3;
create table IF NOT EXISTS mercado_avanzado_calculo_aux3 (id int,tendencia
int);
insert into mercado_avanzado_calculo_aux3 (id,tendencia) select
id,calc*calc2*calc3 from mercado_avanzado_calculo_aux2;

drop table IF EXISTS mercado_avanzado_calculo_aux4;
create table IF NOT EXISTS mercado_avanzado_calculo_aux4 (id int,calc int);
insert into mercado_avanzado_calculo_aux4 select id,
case when tendencia=0 then -1
else 1
End
from mercado_avanzado_calculo_aux3;

drop table IF EXISTS mercado_avanzado_calculo;
create table IF NOT EXISTS mercado_avanzado_calculo AS
select n1.id,n1.dif,n2.calc
from aux_mercado_avanzado_ibex1 n1
inner join mercado_avanzado_calculo_aux4 n2
on n1.id=n2.id;

drop table IF EXISTS aux_resultado_mercado_avanzado;
create table IF NOT EXISTS aux_resultado_mercado_avanzado AS
select dif*calc as aux_resultado
from mercado_avanzado_calculo;

drop table IF EXISTS resultado_mercado_avanzado;
create table IF NOT EXISTS resultado_mercado_avanzado (resultado float);
insert into resultado_mercado_avanzado (resultado) select count(*) from
aux_resultado_mercado_avanzado where aux_resultado>=0;

select (resultado/('${hiveconf:longitud}'-1))*100 from
resultado_mercado_avanzado;
```

5. Uso del sistema

En esta apartado se va a explicar la manera de proceder para poder utilizar el sistema Big Data implementado en el apartado anterior (véase apartado 4), desde el arranque de las máquinas, Hadoop y Hive como el uso de los algoritmos creados para el análisis de datos procedentes de mercados financieros.

5.1. Arranque de las máquinas

El primer paso es el arranque de todas las máquinas que conforman nuestra infraestructura. En caso de tener implementada una solución con una configuración pseudo-distribuida tan solo sería necesario arrancar la máquina que contiene el maestro y el esclavo, pero en el caso de tener una configuración multinodo será necesario arrancar todas y cada una de las máquinas, tanto el maestro como los esclavos.

Una vez arrancadas todas las máquinas hay que hacer dos comprobaciones en las máquinas que actúan como esclavo.

1. Correcta conexión a la red: La máquina tiene que estar convenientemente conectada a la misma red que el resto de máquinas, además, la configuración de red, es decir, la dirección IP, debe estar configurada tal y como indicamos en los ficheros de configuración de Hadoop (véase apartado 4.2.2.3.1).

2. Conexión ssh habilitada: La conexión ssh entre las máquinas esclavo y el maestro debe estar habilitada, permitiendo las comunicaciones en ambos sentidos.

5.2. Arranque de Hadoop

Una vez arrancadas todas las máquinas, nos disponemos a arrancar el sistema Hadoop desde la máquina que ejerce como maestro. Para ello es necesario ejecutar una serie de sentencias [32] en la consola de comandos de linux:

$ start-all.sh (ver Figura 64)

Nuestra instalación de Hadoop nos indica que el comando anterior está en desuso y nos recomienda utilizar los siguientes:

$ start-dfs.sh (ver Figura 65)

Con este comando se inician los demonios HDFS, el namenode y los datanodes.

$ start-yarn.sh (ver Figura 66)

Con este comando se inician todos los procesos pertenecientes al YARN, incluyendo los demonios de Mapreduce.

```
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ start-all.shThis script is
Deprecated. Instead use start-dfs.sh and start-yarn.sh
16/06/15 16:12:22 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Starting namenodes on [HadoopMaster]
hduser@hadoopmaster's password:
HadoopMaster: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-
namenode-HadoopMaster.out
hduser@hadoopslave1's password:
HadoopSlave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-
datanode-HadoopSlave1.out
Starting secondary namenodes [0.0.0.0]
hduser@0.0.0.0's password:
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-
hduser-secondarynamenode-HadoopMaster.out
16/06/15 16:12:54 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-
resourcemanager-HadoopMaster.out
hduser@hadoopslave1's password:
HadoopSlave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-
hduser-nodemanager-HadoopSlave1.out
```

Figura 64: Secuencia de inicio procesos Hadoop desfasado

```
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ start-dfs.sh
16/06/15 16:20:27 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Starting namenodes on [HadoopMaster]
hduser@hadoopmaster's password:
HadoopMaster: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-
namenode-HadoopMaster.out
hduser@hadoopslave1's password:
HadoopSlave1: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-
datanode-HadoopSlave1.out
Starting secondary namenodes [0.0.0.0]
hduser@0.0.0.0's password:
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-
hduser-secondarynamenode-HadoopMaster.out
16/06/15 16:21:09 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

Figura 65: Secuencia de inicio procesos HDFS

```
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-
resourcemanager-HadoopMaster.out
hduser@hadoopslave1's password:
HadoopSlave1: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-
hduser-nodemanager-HadoopSlave1.out
```

Figura 66: Secuencia de inicio procesos YARN

5.2.1. Verificación del correcto funcionamiento de los demonios

Para verificar que el arranque de los demonios ha sido el correcto ejecutamos la sentencia que nos muestra los procesos que están siendo ejecutados:

```
$ jps
```

La lista de procesos que deben estar presentes durante la ejecución del sistema Hadoop varía dependiendo de la configuración:

-Configuración pseudo-distribuida

Lugar	Proceso
Maestro	NodeManager
Maestro	DataNode
Maestro	NameNode
Maestro	ResourceManager
Maestro	SecondaryNameNode
Maestro	Jps

Tabla 6: Ejecución de procesos en configuración pseudo-distribuida

-Configuración multinodo

PROCESO	
Maestro	**Esclavo**
NameNode	NodeManager
ResourceManager	DataNode
SecondaryNameNode	Jps
Jps	

Tabla 7: Ejecución de procesos en configuración multinodo

5.2.2. Gestión del sistema Hadoop a través de interfaz web

Una vez arrancados todos los demonios podemos acceder a la supervisión de la configuración global de Hadoop mediante interfaz web utilizando cualquier navegador, a través de la siguiente dirección:

http://hadoopmaster:50070/dfshealth.html#tab-overview

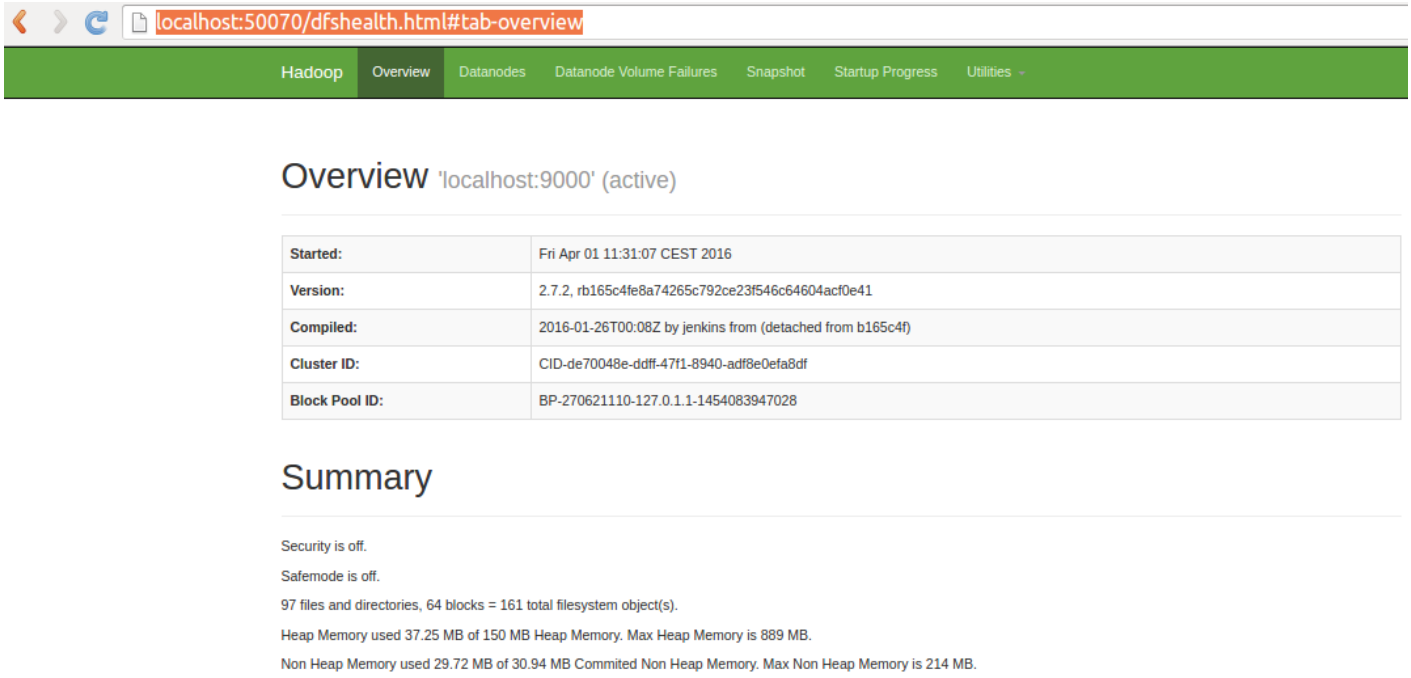


Figura 67: Supervisión sistema Hadoop a través de interfaz web

Además, también podemos acceder a la supervisión de los nodos del cluster:

http://hadoopmaster:8088/cluster

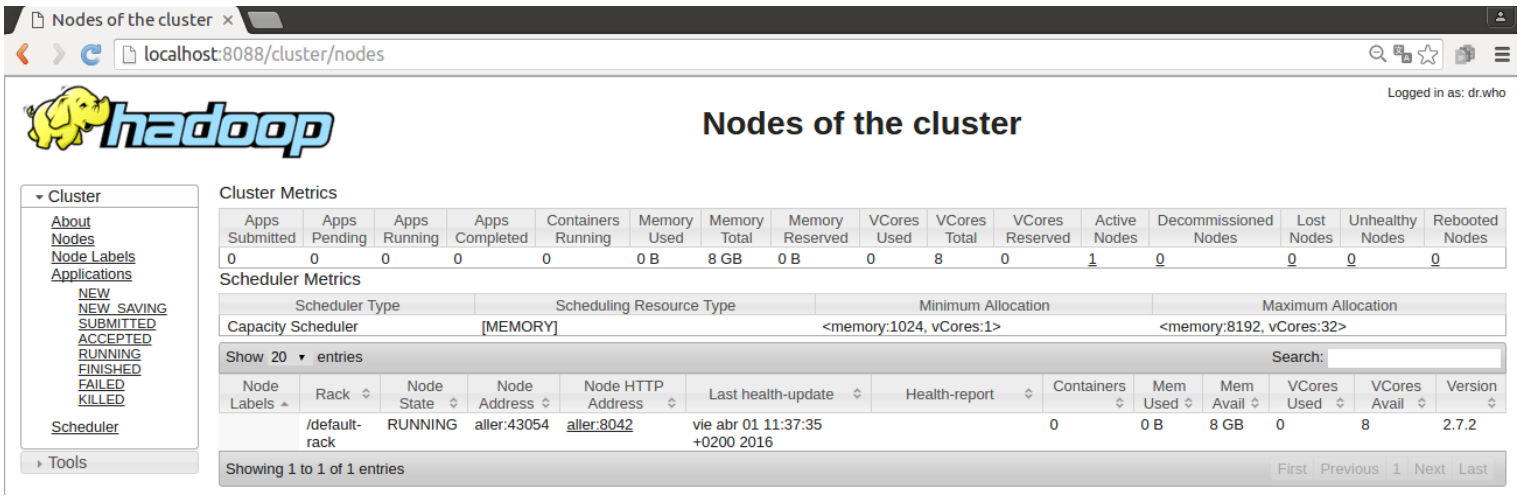


Figura 68: Supervisión nodos del cluster a través de interfaz web

5.3. Arranque de Hive

Para poder arrancar la consola de comandos de Hive para trabajar con instrucciones HiveQL es imprescindible haber arrancado antes Hadoop, una vez iniciadas todas las máquinas y Hadoop se ejecuta la siguiente sentencia para acceder arrancar Hive:

$ hive

```
hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ hive
Logging initialized using configuration in
jar:file:/usr/local/hive/lib/hive-common-1.2.1.jar!/hive-log4j.properties
hive>
```

Figura 69: Secuencia de inicio Hive

5.4. Utilización de scripts HQL

La utilización de Scripts HQL es una manera alternativa de ejecutar sentencias para simplificar tareas repetitivas y agilizar cálculos que requieren de muchos comandos para obtener un resultado.

Para arrancar Hive desde un script HQL es necesario que se realice desde la misma carpeta que contiene al script, y que el fichero tengo una extensión ".hql".El comando a ejecutar para ejecutar un script es el siguiente: **hive -f nombre_del_fichero.hql**

A continuación se muestra la relación de scripts que contienen los diferentes algoritmos generados (véase apartado 4.4) utilizados en este documento:

Script HQL	Descripción
algoritmo_base_nasdaq.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo básico en Nasdaq
algoritmo_base_ftse.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo básico en FTSE
algoritmo_base_nikkei.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo básico en Nikkei
algoritmo_base_ibex35.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo básico en Ibex35
algoritmo_base_simplificado.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo básico simplificado
algoritmo_avanzado.hql	Script que contiene las instrucciones necesarias para ejecutar el algoritmo avanzado

Tabla 8: Listado completo de scripts empleados

5.5. Apagado del sistema

Para finalizar el sistema Hadoop y Hive empleamos las siguientes sentencias:

Sentencia	Utilidad
$ stop-all.sh	Detiene todos los demonios de Hadoop (en desuso)
$ stop-dfs.sh	Detiene los demonios HDFS, el namenode y los datanodes
$ stop-yarn.sh	Detiene los demonios Yarn, mapreduce
Hive: Ctrl + z	Detiene la consola de Hive

Tabla 9: Sentencias para el apagado del sistema

6. Pruebas y resultados

6.1. Introducción

Una vez finalizada la implementación de nuestra arquitectura Big-data, se va a proceder a realizar pruebas sobre escenarios diferentes con el objetivo de encontrar las limitaciones del sistema, así como conocer su escalabilidad. A su vez, también se estudiará la eficiencia de los algoritmos de inversiones financieras generados, en términos económicos.

Para realizar estas pruebas se van a realizar dos tipos de configuraciones, pseudo-distribuida y multinodo, dónde para cada una de ellas se aplicarán los ficheros de datos originales procedentes de los diferentes mercados financieros así como datos sintéticos para conocer el rendimiento de la infraestructura en condiciones de alta carga de datos.

6.2. Entorno de prueba

Para cada tipo de configuración hay un número de máquinas diferente, cada una de ellas con sus características, las cuáles se detallan en los subapartados siguientes:

6.2.1. Configuración pseudo-distribuida

Sólo existe una máquina que ejerce como maestro y esclavo, que cuenta con la siguiente configuración:

General
Procesador: 4x AMD A4-5000 APU with Radeon(TM) HD Graphics Memoria ram: 5543MB Sistema Operativo: Ubuntu 15.10 Usuario: miguelaller (MiguelAller)
Procesadores
AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz AMD A4-5000 APU with Radeon(TM) HD Graphics 1300,00MHz
Almacenamiento
ATA TOSHIBA MQ01ABF0
Red
Controlador Ethernet: Qualcomm Atheros AR8162 Fast Ethernet Controlador de red: Realtek Semiconductor Co., Ltd. RTL8188EE Wireless Network Adapter

Tabla 10: Especificaciones maestro en sistema pseudo-distribuido

6.2.2. Cluster doméstico

En esta configuración se cuenta con diferentes combinaciones de máquinas:

- 1 nodo: Un maestro y un esclavo
- 2 nodos: Un maestro y dos esclavos
- 3 nodos: Un maestro y tres esclavos

En total se emplean cuatro máquinas, cada una con su propia configuración:

Maestro: Misma configuración que la expuesta en la Tabla 10

Esclavo 1:

General
Procesador: 4x Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz Memoria ram: 4046MB Sistema Operativo: Ubuntu 15.10 Usuario: miguelaller (MiguelAller)
Procesadores
Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2000,00MHz Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2000,00MHz Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2333,00MHz Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2333,00MHz
Almacenamiento
ATA WDC WD10EADS-65L
Red
Controlador Ethernet: Realtek Semiconductor. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller Controlador de red: Ralink corp. RT2790 Wireless 802.11n 1T/2R PCIe

Tabla 11: Especificaciones esclavo 1 en cluster doméstico

Esclavo 2:

General
Procesador: 4x Intel(R) Core(TM) i5 CPU M 460 @ 2.53GHz Memoria ram: 3841MB Sistema Operativo: Ubuntu 15.10 Usuario: miguelaller (MiguelAller)
Procesadores
Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2534,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 1733,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2534,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2399,00MHz

Almacenamiento
ATA SAMSUNG HM641JI
Red
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller Controlador de red: Ralink corp. RT2790 Wireless 802.11n 1T/2R PCIe

Tabla 12: Especificaciones esclavo 2 en cluster doméstico

Esclavo 3:

General
Procesador: 4x Intel(R) Core(TM) i3 CPU M 460 @ 1.83GHz Memoria ram: 4386 MB Sistema Operativo: Ubuntu 15.10 Usuario: hduser ()
Procesadores
Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1500,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1500,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1630,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1630,00MHz
Almacenamiento
ATA WDC WD5000AAKS-6
Red
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL8101/2/6E PCI Express Fast/Gigabit Ethernet controller Controlador de red: Ralink corp. RT3092 Wireless 802.11n 2T/2R PCIe

Tabla 13: Especificaciones esclavo 3 en cluster doméstico

Como se puede observar en las tablas anteriores tanto el maestro como los esclavos son equipos de cuatro núcleos y en torno a 4 GB de memoria ram, garantizando una homogeneidad en la configuración, pero no creando un sistema ideal, aportando cada equipo aproximadamente los mismos recursos al sistema.

6.2.3. Cluster Universitario

En esta configuración, todas las máquinas empleadas son iguales, presentado los mismos recursos, creando un cluster ideal por la homogeneidad entre equipos. Dicha configuración es la siguiente:

General
Procesador: 4x Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz Memoria Ram: 3310MB Sistema Operativo: Debian GNU/Linux 8.1 Usuario: 0284736 (GRADO EN INGENIERIA DE SISTEMAS DE COMUNICACIONES)
Procesadores
Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz
Almacenamiento
ATA WDC WD5000AAKX-0 TSSTcorp CDDVDW SH-224DB
Red
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL-8100/8101L/8139 PCI Fast Ethernet

Tabla 14: Especificaciones equipos en cluster universitario

6.3. Efectividad de los algoritmos generados

6.3.1. Efectividad del algoritmo básico

El objetivo final del algoritmo implementado es la de conseguir una herramienta de predicción fiable, es por ello que el porcentaje de acierto del mismo es importante. A continuación se muestran los resultados obtenidos para los diferentes mercados financieros que son objeto de nuestro análisis:

Algoritmo	Mercado	Nº Datos	Nº Aciertos	Nº Errores	% de acierto
algoritmo_base_simplificado.hql	Nasdaq	11388	7603	3784	66.77 %
algoritmo_base_simplificado.hql	Ftse	8408	4269	4138	50.78 %
algoritmo_base_simplificado.hql	Nikkei	7949	4162	3786	52.37 %
algoritmo_base_simplificado.hql	Ibex 35	5813	2918	2894	50.21 %

Tabla 15: Efectividad algoritmo básico de inversión para mercados financieros

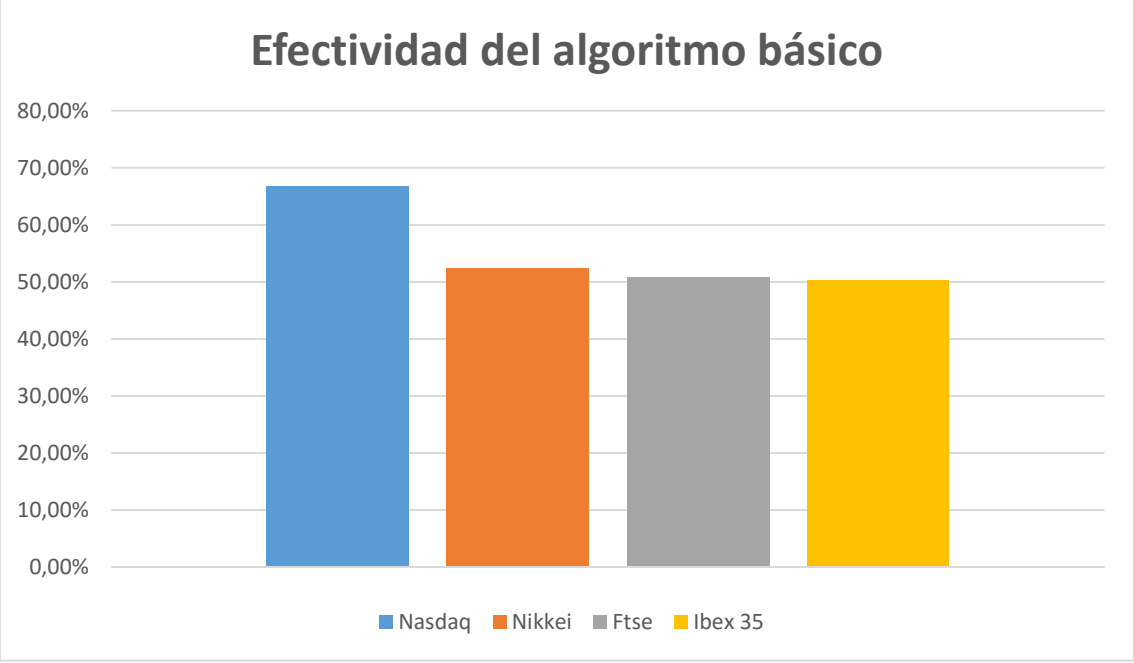


Figura 70: Efectividad algoritmos de inversión básico

Como vemos en la Tabla 15 y en la Figura 70 , el mercado que presenta unos mejores resultados es el Nasdaq, mercado en el que contamos con una mayor cantidad de datos, mientras que el mercado que presenta los peores resultados es el Ibex 35, del cual tenemos una menor cantidad de datos. Al ser una muestra de mercados menor no se puede extrapolar los resultados a una normal general, pero es evidente que a mayor

cantidad de datos mejor podremos predecir los acontecimientos futuros, ya que el comportamiento histórico de los mercados sigue patrones comunes en determinados periodos de tiempo.

6.3.2. Efectividad del algoritmo avanzado

A continuación se muestran los resultados obtenidos para el Ibex a partir de la combinación de datos de diferentes mercados:

Algoritmo	Nº Datos	Nº Datos Procesados	Nº Aciertos	Nº Errores	% de acierto
algoritmo_avanzado.hql	Ibex Nasdaq Nikkei	23252	2893	2920	49.77 %

Tabla 16: Efectividad algoritmo de inversión avanzado

A diferencia de lo que cabría esperar al combinar datos de diferentes mercados financieros, la efectividad obtenida es menor comparada con la del algoritmo básico Sin embargo, haciendo un análisis del mercado Japonés (Nikkei) podemos observar que es una economía que se encuentra en continuas recesiones [33], hecho que afecta directamente a nuestra predicción, haciendo que la tendencia bajista del Nikkei arrastre errores hasta nuestra predicción de la tendencia del Ibex 35.

6.4. Eficiencia de la infraestructura

6.4.1. Introducción

Una vez hemos obtenido el rendimiento de los algoritmos generados, vamos a proceder a estudiar el rendimiento de nuestra infraestructura Hadoop a partir de diferentes configuraciones, aplicando los algoritmos generados para generar carga de datos al sistema.

En los apartados siguientes se muestra el rendimiento para la configuración pseudo-distribuida y para la configuración multinodo, aplicando los algoritmos básico y avanzado a partir de los datos obtenidos en el apartado 4.3 y además, a partir de datos sintéticos.

6.4.2. Datos sintéticos

Creamos los datos sintéticos con el objetivo de analizar los límites de la infraestructura, ya que no disponemos de tanta cantidad de datos a partir de las fuentes empleadas.

Incrementando el número de datos por dos en cada iteración nos permite obtener un numero de datos lo suficientemente grande como para ver la relación existente entre el número de datos y tiempo de procesamiento.

Esta parte del análisis no tiene validez en cuanto a los datos obtenidos con el algoritmo como si sucedía en los apartados anteriores. El objetivo en este caso es el del estudio del tiempo empleado por la infraestructura en el análisis de los datos sintéticos.

Para realizar el citado análisis vamos a utilizar el mercado NASDAQ ya que es el que cuenta con un mayor número de datos. Multiplicando el número de valores por dos obtenemos:

Nº de datos	Tamaño del fichero	Nombre del fichero
0	41B	Nasdaq_0.csv
100	8,1 kB	Nasdaq_1.csv
200	16,2 kB	Nasdaq_2.csv
400	32,4 kB	Nasdaq_3.csv
800	64,6 kB	Nasdaq_4.csv
1600	128,9 kB	Nasdaq_5.csv
3200	257,4 kB	Nasdaq_6.csv
6400	506,3 kB	Nasdaq_7.csv
12800	954,4 kB	Nasdaq_8.csv
25600	1,9 MB	Nasdaq_9.csv
51200	3,8 MB	Nasdaq_10.csv
102400	7,6 MB	Nasdaq_11.csv
204800	15,3 MB	Nasdaq_12.csv
409600	30,5 MB	Nasdaq_13.csv
819200	61,1 MB	Nasdaq_14.csv
1638400	123,8 MB	Nasdaq_15.csv
3276800	244,3 MB	Nasdaq_16.csv
6553600	488,6 MB	Nasdaq_17.csv

Tabla 17: Listado de archivos generados a partir de datos sintéticos

El primer valor de la tabla anterior es 0 para comprobar el tiempo de procesamiento del sistema Hive sin datos, de esta manera podemos establecer que parte del tiempo de procesamiento es causado por la propia arquitectura y que parte proviene de los datos.

Además, el tamaño de ese fichero no es 0 pese a no contener ningún dato debido a las cabeceras de cada columna.

El procedimiento para la creación de los ficheros detallados en la tabla es muy sencillo, duplicando por dos el archivo original, obteniendo en cada paso un fichero con el doble de datos que el anterior.

Para el análisis de los nuevos ficheros generados sintéticamente vamos a utilizar el algoritmo simplificado ya que nos permite una mayor rapidez de ejecución entre un fichero y otro, teniendo que cambiar tan solo la variable longitud y el fichero desde donde se cargan los datos, de la siguiente manera:

algoritmo_base_simplificado_datos_sintéticos.hql

```
set longitud=(longitud del fichero a analizar);

create database IF NOT EXISTS basedatos_financiera_sinteticos;
use basedatos_financiera_sinteticos;

create table IF NOT EXISTS mercado_sinteticos (fecha string,open float,high
float,low float,close float,volume float,adjclose float) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/(mercado a
analizar).csv' overwrite into table mercado_sinteticos;

drop table IF EXISTS aux_mercado_sinteticos;
create table IF NOT EXISTS aux_mercado_sinteticos (id int,dif float);
insert into aux_mercado_sinteticos (id,dif) select row_number() over(),close-
open from mercado_sinteticos;

drop table IF EXISTS aux_mercado_sinteticos1;
create table IF NOT EXISTS aux_mercado_sinteticos1 (id int,dif float);
drop table IF EXISTS aux_mercado_sinteticos2;
create table IF NOT EXISTS aux_mercado_sinteticos2 (id int,predic float);

insert into aux_mercado_sinteticos1 (id,dif) select row_number() over(),dif
from aux_mercado_sinteticos where aux_mercado_sinteticos.id >=1   and
aux_mercado_sinteticos.id <='${hiveconf:longitud}'-1;

insert into aux_mercado_sinteticos2 (id,predic) select row_number()
over(),dif from aux_mercado_sinteticos where aux_mercado_sinteticos.id >=2
and  aux_mercado_sinteticos.id <='${hiveconf:longitud}';

drop table IF EXISTS mercado_sinteticos_calculo;
create table IF NOT EXISTS mercado_sinteticos_calculo AS
select n1.dif,n2.predic
from aux_mercado_sinteticos1 n1
inner join aux_mercado_sinteticos2 n2
on n1.id=n2.id;
```

```
drop table IF EXISTS aux_mercado_sinteticos_calculo;
create table IF NOT EXISTS aux_mercado_sinteticos_calculo AS
select dif*predic as aux_resultado
from mercado_sinteticos_calculo;

drop table IF EXISTS resultado_mercado_sinteticos;
create table IF NOT EXISTS resultado_mercado_sinteticos (resultado float);
insert into resultado_mercado_sinteticos (resultado) select count(*) from
aux_mercado_sinteticos_calculo where aux_resultado>=0;

select (resultado/('${hiveconf:longitud}'-1))*100 from
resultado_mercado_sinteticos;
```

Figura 71: Adaptación del algoritmo básico simplificado para datos sintéticos

6.4.3. Modo pseudo-distribuido

Vamos a comenzar el estudio de la eficiencia de la infraestructura Hadoop con la configuración pseudo-distribuida, en dónde disponemos de una sola máquina que actúa al mismo tiempo como maestro y cómo esclavo.

6.4.3.1. Análisis con datos procedentes mercados financieros

Tiempos de procesamiento del algoritmo básico

Para contabilizar el tiempo que tarda el script creado se usa el siguiente comando:

time hive -f algoritmo_base_(nombre del mercado que queremos medir).hql

Los tiempos de procesamiento obtenidos para los diferentes mercados financieros son los siguientes:

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	4m50.258s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	4m50.748s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	4m47.065s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	4m44.703s

Tabla 18: Tiempo de procesamiento algoritmo básico sistema pseudo-distribuido

Como se puede observar en la Tabla 18, el número de datos apenas afecta al tiempo de procesamiento, ya que se trata de una cantidad de datos pequeña y el arranque del sistema Hadoop, así como los procesos de Mapreduce son los que afectan directamente

a ese tiempo. Esto se verá de una manera muy clara en el apartado 6.4.3.2 al utilizar datos sintéticos que nos proporcionan una visión de conjunto del problema.

Tiempo de procesamiento del algoritmo avanzado

Para contabilizar el tiempo que tarda el algoritmo avanzado usamos el siguiente comando:

time hive -f algoritmo_avanzado.hql

En este caso, el tiempo de procesamiento sólo es uno, ya que el propio algoritmo combina los diferentes mercados creando un solo archivo interno en Hadoop con todos los datos analizados.

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	5m3.531s

Tabla 19: Tiempo de procesamiento algoritmo avanzado sistema pseudo-distribuido

El tiempo de procesamiento es más elevado en este caso que utilizando el algoritmo básico, esto es debido principalmente a dos motivos:

-El número de datos procesados es mayor, doblando el del caso peor del algoritmo básico, el cual contaba con 11388 datos.

-El número de ficheros empleados es mayor, recogiendo datos de tres tablas, haciendo más operaciones de lectura.

-Al utilizar datos procedentes de tres tablas de datos diferentes para combinarlas y obtener un resultado, las operaciones realizadas en Hive de creación y unión de tablas requieren un mayor esfuerzo computacional.

6.4.3.2. Análisis con datos de mercados sintéticos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	5m15.879s	0
100	Nasdaq_1.csv	5m16.275s	0,31
200	Nasdaq_2.csv	5m25.790s	0,61
400	Nasdaq_3.csv	5m22.563s	1,24
800	Nasdaq_4.csv	5m20.313s	2,49
1600	Nasdaq_5.csv	5m29.341s	4,85
3200	Nasdaq_6.csv	5m39.005s	9,44
6400	Nasdaq_7.csv	5m43.531s	18,63
12800	Nasdaq_8.csv	5m55.236s	36,03
25600	Nasdaq_9.csv	5m59.000s	71,31
51200	Nasdaq_10.csv	6m4.942s	140,30
102400	Nasdaq_11.csv	6m9.371s	277,22
204800	Nasdaq_12.csv	6m25.630s	531,08
409600	Nasdaq_13.csv	6m56.382s	983,71
819200	Nasdaq_14.csv	7m32.533s	1810,25
1638400	Nasdaq_15.csv	9m44.362s	2803,74
3276800	Nasdaq_16.csv	14m0.176s	3900,13
6553600	Nasdaq_17.csv	No finaliza la ejecución	0

Tabla 20: Tiempos de procesamiento datos sintéticos sistema pseudo-distribuido

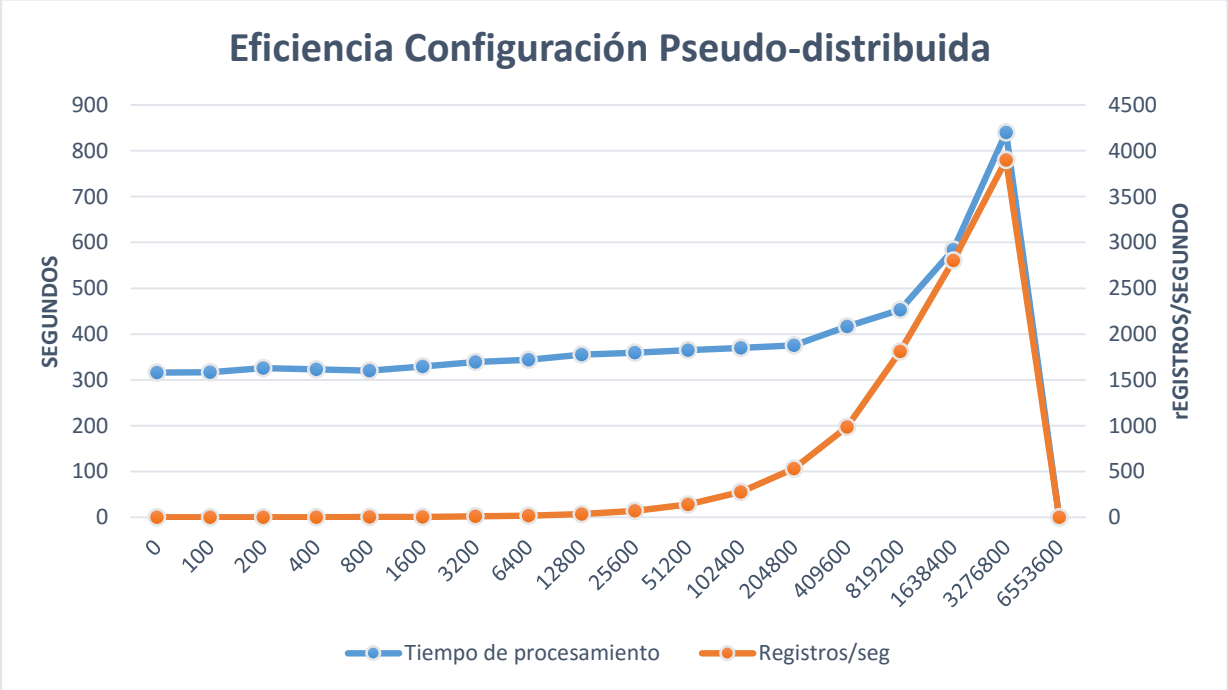


Figura 72: Eficiencia configuración pseudo-distribuida

En la Figura 72 se puede observar cómo se mantiene constante el tiempo de procesamiento hasta los 51200 datos, dónde comienza a subir y comienza a pegar saltos bruscos, hasta llegar a los 6553600 dónde el proceso no finaliza por falta de memoria ram del sistema.

El número de registros procesados por segundo sigue la tendencia del tiempo de procesamiento, obteniendo una eficiencia mayor cuántos más datos recibe el sistema, hasta llegar al punto dónde ya no se finaliza el proceso.

6.4.4. Cluster doméstico

Se va a realizar el mismo procedimiento que en el apartado anterior, pero esta vez para una configuración multinodo basada en un cluster doméstico, dónde hay un maestro y tres esclavos, todos ellos en diferentes máquinas. Una vez ha sido iniciado el sistema Hadoop, podemos ver el funcionamiento de los nodos configurados, tal y como se muestra en la Figura 73.

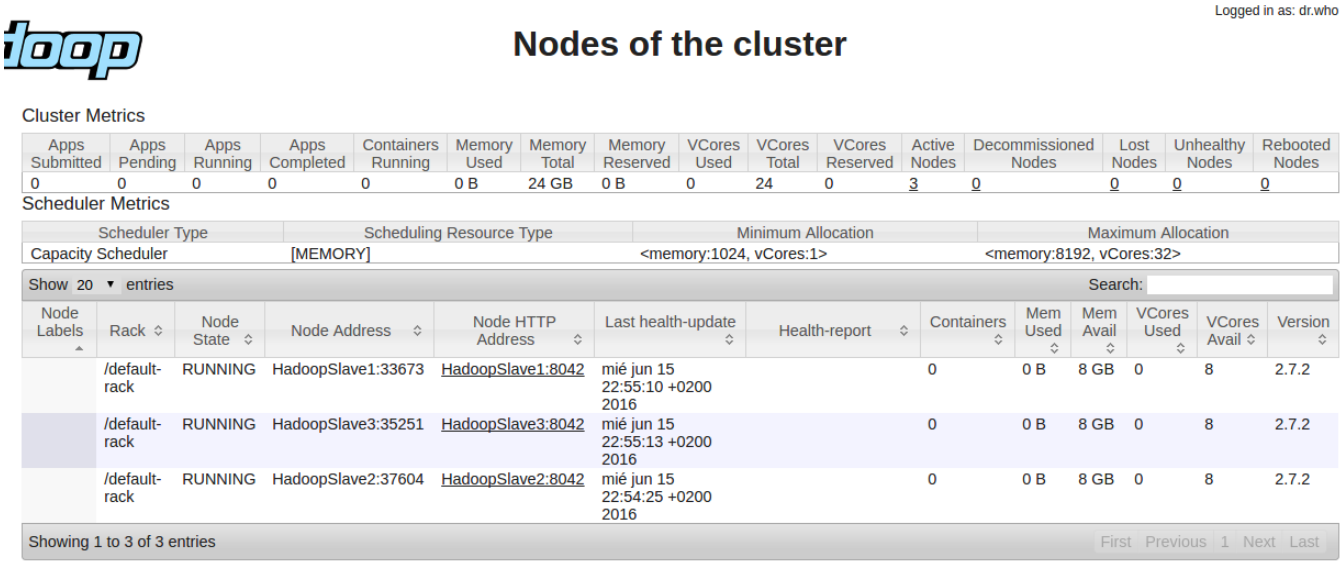


Figura 73: Nodos activos en el cluster doméstico durante la ejecución de Hadoop

6.4.4.1. Análisis con datos de mercados financieros

Los tiempos de procesamiento obtenidos para los diferentes mercados financieros son los expuestos a continuación.

6.4.4.1.1. Un nodo

Tiempos de procesamiento del algoritmo básico

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m32.548s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	1m2.692s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	0m57.472s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	0m55.434s

Tabla 21: Tiempo de procesamiento algoritmo básico cluster doméstico (1 nodo)

Tiempos de procesamiento del algoritmo avanzado

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	1m57.353s

Tabla 22: Tiempo de procesamiento algoritmo avanzado cluster doméstico (1 nodo)

6.4.4.1.2. Dos nodos

Tiempos de procesamiento del algoritmo básico

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m4.153s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	1m0.523s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	1m0.606s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	1m2.584s

Tabla 23: Tiempo de procesamiento algoritmo básico cluster doméstico (2 nodos)

Tiempos de procesamiento del algoritmo avanzado

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	2m2.940s

Tabla 24: Tiempo de procesamiento algoritmo avanzado cluster doméstico (2 nodos)

6.4.4.1.3. Tres nodos

Tiempos de procesamiento del algoritmo básico

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m4.641s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	0m57.628s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	0m58.165s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	0m56.843s

Tabla 25: Tiempo de procesamiento algoritmo básico cluster doméstico (3 nodos)

Tiempos de procesamiento del algoritmo avanzado

Algoritmo	Tamaño Fichero	Mercado	Nº Datos	Tiempo procesamiento
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	1m50.098s

Tabla 26: Tiempo de procesamiento algoritmo avanzado cluster doméstico (3 nodos)

Al igual que sucedía en la configuración pseudo-distribuida, el tiempo de procesamiento no se ve afectado por el número de datos, al ser cantidades de datos pequeñas. Cabe destacar el peor rendimiento para dos nodos que para uno y tres nodos, estudiando las posibles causas de este fenómeno al finalizar el análisis completo.

6.4.4.2. Análisis con datos de mercados sintéticos

6.4.4.2.1. Un nodo esclavo

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m55.629s	0
100	Nasdaq_1.csv	0m57.993s	1,72
200	Nasdaq_2.csv	0m59.707s	3,35
400	Nasdaq_3.csv	0m59.383s	6,74
800	Nasdaq_4.csv	0m59.737s	13,4
1600	Nasdaq_5.csv	1m0.358s	26,65
3200	Nasdaq_6.csv	1m2.223s	51,42
6400	Nasdaq_7.csv	1m3.519s	100,75
12800	Nasdaq_8.csv	1m4.369s	198,85
25600	Nasdaq_9.csv	1m5.404s	391,41
51200	Nasdaq_10.csv	1m12.410s	707,02
102400	Nasdaq_11.csv	1m19.677s	1285,18
204800	Nasdaq_12.csv	1m41.244s	2022,83
409600	Nasdaq_13.csv	2m42.269s	2524,203
819200	Nasdaq_14.csv	4m55.067s	2776,32
1638400	Nasdaq_15.csv	6m31.814s	4181,57
3276800	Nasdaq_16.csv	12m58.855s	4207,20
6553600	Nasdaq_17.csv	23m49.937s	4583,14

Tabla 27: Tiempos de procesamiento datos sintéticos cluster doméstico (1 nodo)

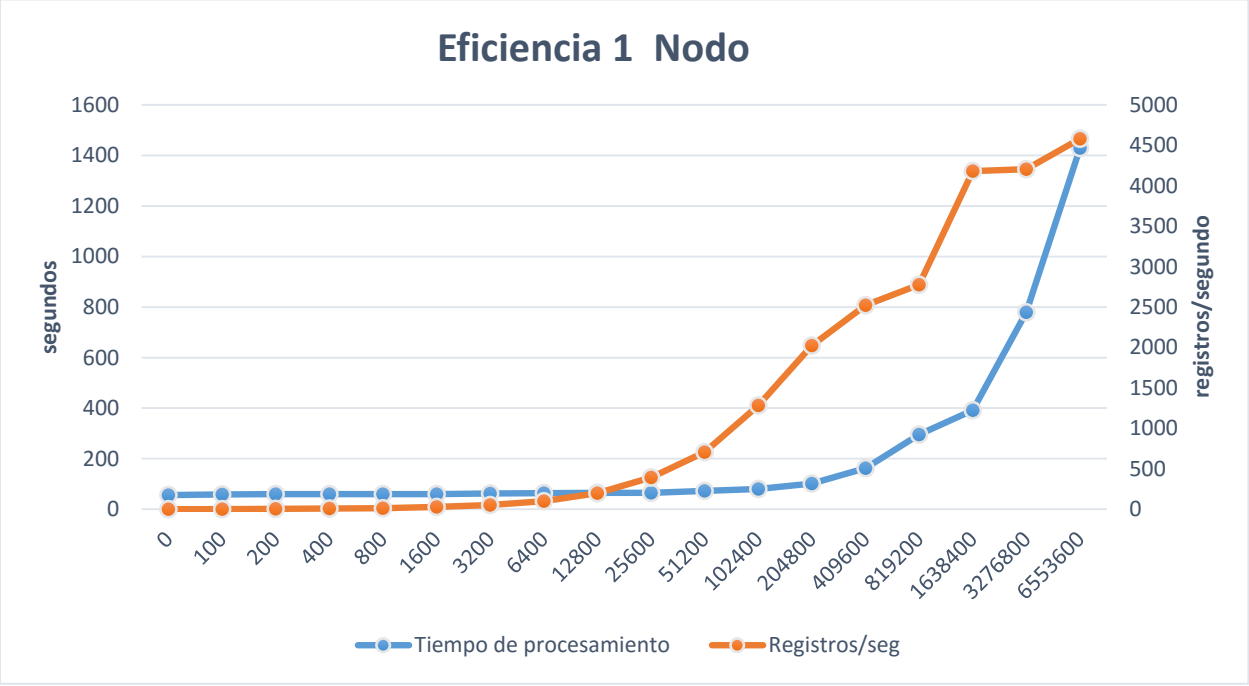


Figura 74: Eficiencia configuración multinodo (1 nodo) - cluster doméstico

6.4.4.2.2. Dos nodos esclavos

Nº de datos	Nombre del fichero	Tiempo de procesamiento	Registros/Segundo
0	Nasdaq_0.csv	1m0.524s	0
100	Nasdaq_1.csv	1m4.624s	1,54
200	Nasdaq_2.csv	1m7.160s	2,97
400	Nasdaq_3.csv	1m2.778s	6,37
800	Nasdaq_4.csv	1m3.152s	12,66
1600	Nasdaq_5.csv	1m2.143s	25,74
3200	Nasdaq_6.csv	1m6.148s	48,37
6400	Nasdaq_7.csv	1m8.220s	93,81
12800	Nasdaq_8.csv	1m8.232s	187,6
25600	Nasdaq_9.csv	1m7.224s	380,81
51200	Nasdaq_10.csv	1m15.465s	678,46
102400	Nasdaq_11.csv	1m25.563s	1196,78
204800	Nasdaq_12.csv	1m48.988s	1879,105
409600	Nasdaq_13.csv	2m35.186s	2639,41
819200	Nasdaq_14.csv	3m34.326s	3822,21
1638400	Nasdaq_15.csv	6m15.112s	4367,76
3276800	Nasdaq_16.csv	14m18.578s	3816,54
6553600	Nasdaq_17.csv	24m41.552s	4423,47

Tabla 28: Tiempos de procesamiento datos sintéticos cluster doméstico (2 nodos)

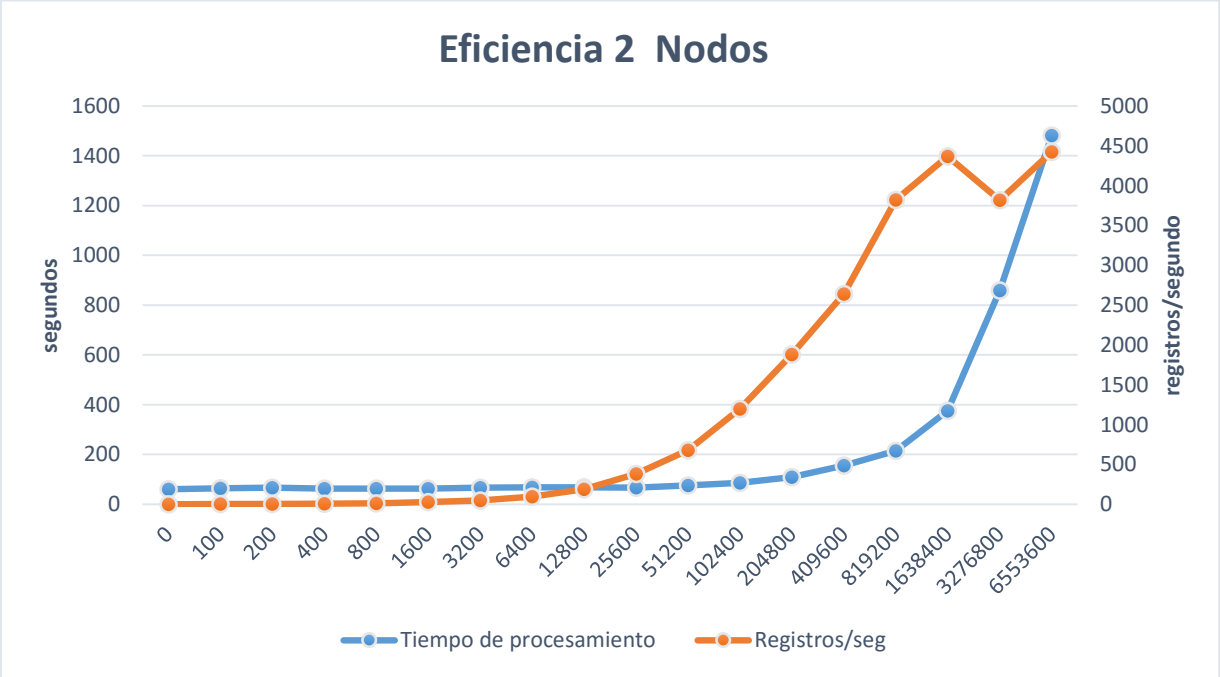


Figura 75: Eficiencia configuración multinodo (2 nodos) – cluster doméstico

6.4.4.2.3. Tres nodos esclavos

Nº de datos	Nombre del fichero	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m57.430s	0
100	Nasdaq_1.csv	0m58.172s	1,72
200	Nasdaq_2.csv	0m58.703s	3,40
400	Nasdaq_3.csv	0m58.339s	6,85
800	Nasdaq_4.csv	0m58.410s	13,7
1600	Nasdaq_5.csv	0m58.069s	27,55
3200	Nasdaq_6.csv	0m57.653s	55,5
6400	Nasdaq_7.csv	1m0.234s	106,62
12800	Nasdaq_8.csv	1m0.617s	213,12
25600	Nasdaq_9.csv	1m3.764s	401,48
51200	Nasdaq_10.csv	1m7.333s	760,4
102400	Nasdaq_11.csv	1m12.957s	1403,56
204800	Nasdaq_12.csv	1m20.777s	2535,37
409600	Nasdaq_13.csv	1m41.254s	4045,27
819200	Nasdaq_14.csv	2m21.537s	5787,88
1638400	Nasdaq_15.csv	3m44.568s	7295,78
3276800	Nasdaq_16.csv	7m1.354s	7776,83
6553600	Nasdaq_17.csv	12m55.394s	8451,96

Tabla 29: Tiempos de procesamiento datos sintéticos cluster doméstico (3 nodos)

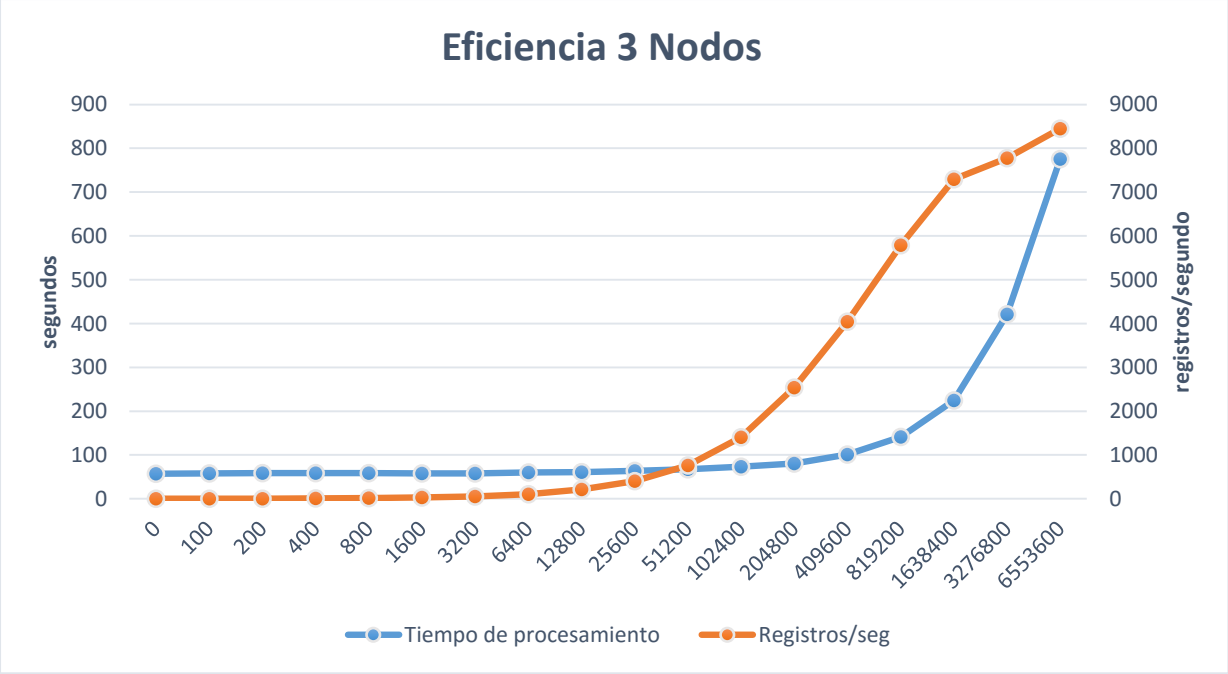


Figura 76: Eficiencia configuración multinodo (3 nodos)- cluster doméstico

6.4.5. Cluster universitario

Se va a realizar el mismo procedimiento que en el apartado anterior, pero esta vez para una configuración multinodo basada en un cluster universitario, dónde hay un maestro y hasta 24 esclavos, todos ellos en diferentes máquinas. Una vez arrancado el sistema Hadoop, la configuración de los nodos queda de la siguiente manera:

Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0 B	192 GB	0 B	0	192	0	24	0	0	0	0

trics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
duler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

ntries Search:

Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack	RUNNING	it003.lab.it.uc3m.es:56213	it003.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	doc026.lab.it.uc3m.es:58702	doc026.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it014.lab.it.uc3m.es:52865	it014.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm006.lab.it.uc3m.es:45624	lm006.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it010.lab.it.uc3m.es:50011	it010.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it007.lab.it.uc3m.es:52930	it007.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it011.lab.it.uc3m.es:46619	it011.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it020.lab.it.uc3m.es:32769	it020.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it017.lab.it.uc3m.es:60462	it017.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm008.lab.it.uc3m.es:44031	lm008.lab.it.uc3m.es:8042	vie jun 17 19:39:47 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it016.lab.it.uc3m.es:55966	it016.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm010.lab.it.uc3m.es:45439	lm010.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it002.lab.it.uc3m.es:51516	it002.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it012.lab.it.uc3m.es:60649	it012.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it005.lab.it.uc3m.es:47216	it005.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it019.lab.it.uc3m.es:53957	it019.lab.it.uc3m.es:8042	vie jun 17 19:39:55 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm001.lab.it.uc3m.es:44105	lm001.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it009.lab.it.uc3m.es:56852	it009.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it008.lab.it.uc3m.es:49346	it008.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it013.lab.it.uc3m.es:49490	it013.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm009.lab.it.uc3m.es:46873	lm009.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it015.lab.it.uc3m.es:59819	it015.lab.it.uc3m.es:8042	vie jun 17 19:39:46 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it018.lab.it.uc3m.es:42131	it018.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it006.lab.it.uc3m.es:53930	it006.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2

4 of 24 entries First Previous 1 Next Last

Figura 77: Nodos activos en cluster universitario durante la ejecución de Hadoop

Además, la configuración de este entorno se ha realizado utilizando una serie de Scripts y realizando conexiones ssh a todas las máquinas implicadas. Dicho procedimiento se encuentra en el anexo II

En esta ocasión, vamos a analizar la eficiencia del entorno, únicamente a partir de los datos sintéticos.

Como limitación del entorno del cluster universitario nos encontramos con la limitación del uso de disco de la cuenta proporcionada por el departamento de telemática, por ello, los archivos Nasdaq_16.csv y Nasdaq_17.csv no han podido ser analizados en ninguno de los casos. Dicha limitación es de 2,8 Gb en cuota blanda y de 3,2 Gb en cuota dura durante siete días.

6.4.5.1. 2 nodos esclavos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m23.987s	0
100	Nasdaq_1.csv	0m24.024s	4,16
200	Nasdaq_2.csv	0m25.221s	7,93
400	Nasdaq_3.csv	0m26.128s	15,31
800	Nasdaq_4.csv	0m24.407s	32,77
1600	Nasdaq_5.csv	0m24.035s	66,57
3200	Nasdaq_6.csv	0m24.383s	131,24
6400	Nasdaq_7.csv	0m23.843s	268,42
12800	Nasdaq_8.csv	0m24.946s	513,11
25600	Nasdaq_9.csv	0m25.109s	1019,55
51200	Nasdaq_10.csv	0m25.143s	2036,35
102400	Nasdaq_11.csv	0m27.417s	3734,91
204800	Nasdaq_12.csv	0m29.807s	6870,87
409600	Nasdaq_13.csv	0m35.837s	11429,52
819200	Nasdaq_14.csv	0m48.035s	17054,23
1638400	Nasdaq_15.csv	1m6.586s	24605,77
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0

Tabla 30: Tiempos de procesamiento datos sintéticos cluster universitario (1 nodo)

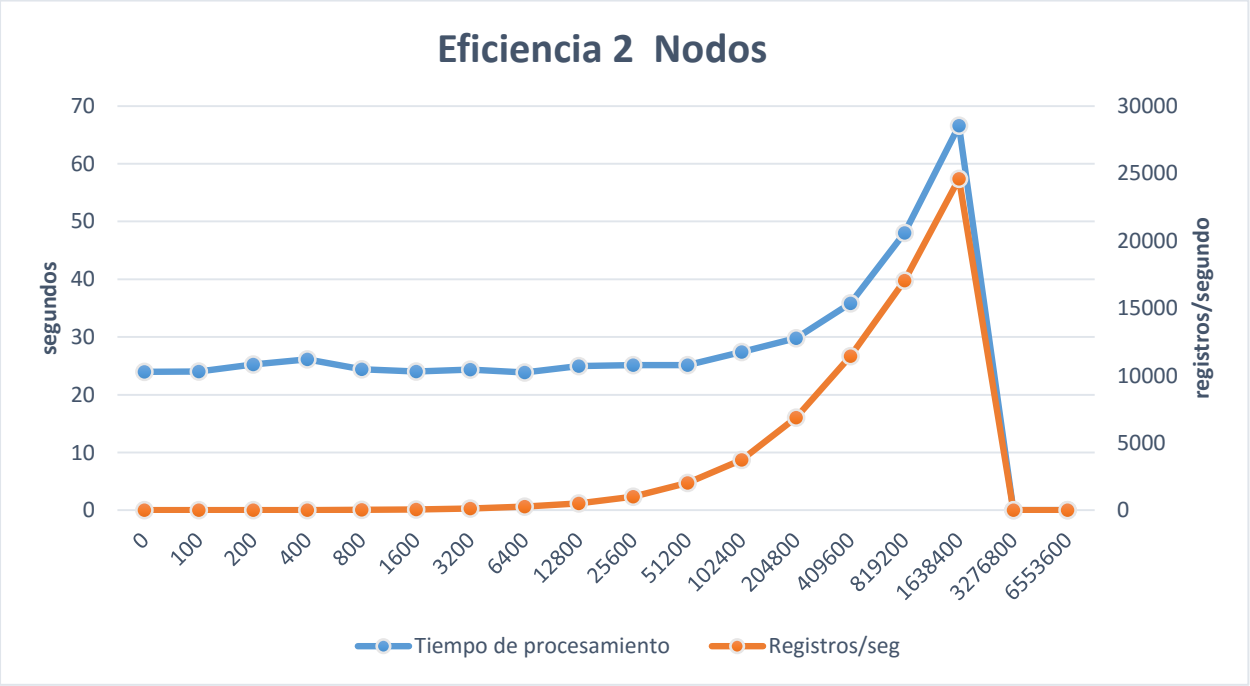


Figura 78: Eficiencia configuración multinodo (2 nodos) – cluster universitario

6.4.5.2. 4 nodos esclavos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m23.469s	0
100	Nasdaq_1.csv	0m24.075s	4,15
200	Nasdaq_2.csv	0m23.921s	8,36
400	Nasdaq_3.csv	0m24.694s	16,2
800	Nasdaq_4.csv	0m25.598s	31,25
1600	Nasdaq_5.csv	0m23.604s	67,78
3200	Nasdaq_6.csv	0m23.822s	134
6400	Nasdaq_7.csv	0m23.773s	269,22
12800	Nasdaq_8.csv	0m25.440s	503,14
25600	Nasdaq_9.csv	0m25.015s	1023,38
51200	Nasdaq_10.csv	0m26.963s	1898,89
102400	Nasdaq_11.csv	0m27.157s	3770,66
204800	Nasdaq_12.csv	0m31.207s	6562,63
409600	Nasdaq_13.csv	0m35.500s	11538,02
819200	Nasdaq_14.csv	0m45.706s	17923,24
1638400	Nasdaq_15.csv	1m4.974s	25216,24
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0

Tabla 31: Tiempos de procesamiento datos sintéticos cluster universitario (4 nodos)

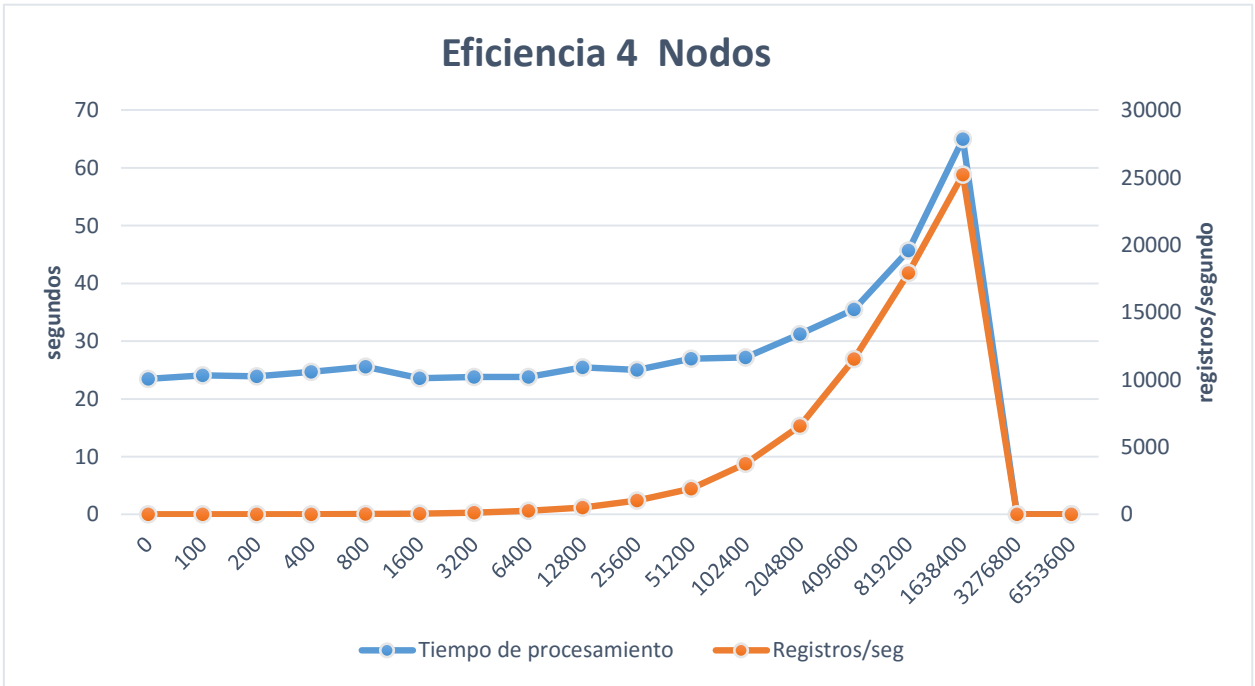


Figura 79: Eficiencia configuración multinodo (4 nodos) – cluster universitario

6.4.5.3. 8 nodos esclavos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m23.614s	0
100	Nasdaq_1.csv	0m23.785s	4,20
200	Nasdaq_2.csv	0m23.508s	8,50
400	Nasdaq_3.csv	0m23.737s	16,85
800	Nasdaq_4.csv	0m24.529s	32,61
1600	Nasdaq_5.csv	0m24.553s	65,16
3200	Nasdaq_6.csv	0m25.226s	126,85
6400	Nasdaq_7.csv	0m24.195s	264,51
12800	Nasdaq_8.csv	0m23.939s	534,69
25600	Nasdaq_9.csv	0m25.362s	1009,38
51200	Nasdaq_10.csv	0m27.072s	1891,25
102400	Nasdaq_11.csv	0m27.334s	3746,25
204800	Nasdaq_12.csv	0m31.704s	6459,75
409600	Nasdaq_13.csv	0m35.697s	11474,35
819200	Nasdaq_14.csv	0m47.957s	17081,97
1638400	Nasdaq_15.csv	1m5.891s	24865,30
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0

Tabla 32: Tiempos de procesamiento datos sintéticos cluster universitario (8 nodos)

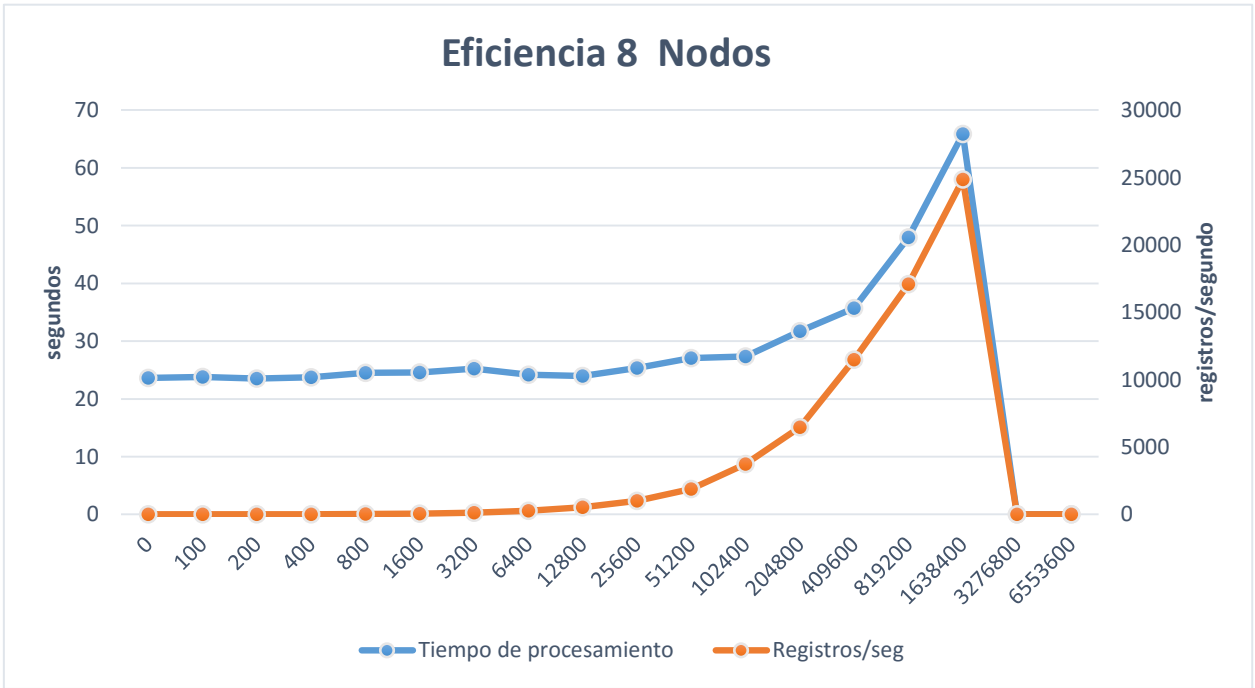


Figura 80: Eficiencia configuración multinodo (8 nodos) – cluster universitario

6.4.5.4. 16 nodos esclavos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m23.592s	0
100	Nasdaq_1.csv	0m23.652s	4,22
200	Nasdaq_2.csv	0m23.225s	8,61
400	Nasdaq_3.csv	0m23.890s	16,74
800	Nasdaq_4.csv	0m26.049s	30,71
1600	Nasdaq_5.csv	0m26.410s	60,58
3200	Nasdaq_6.csv	0m26.651s	120,07
6400	Nasdaq_7.csv	0m25.696s	249,066
12800	Nasdaq_8.csv	0m25.541s	501,15
25600	Nasdaq_9.csv	0m26.438s	968,30
51200	Nasdaq_10.csv	0m27.157s	1885,33
102400	Nasdaq_11.csv	0m27.762s	3688,49
204800	Nasdaq_12.csv	0m31.106s	6583,93
409600	Nasdaq_13.csv	0m35.453s	11553,32
819200	Nasdaq_14.csv	0m46.345s	17676,12
1638400	Nasdaq_15.csv	1m18.814s	20788,18
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0

Tabla 33: Tiempos de procesamiento datos sintéticos cluster universitario (16 nodos)

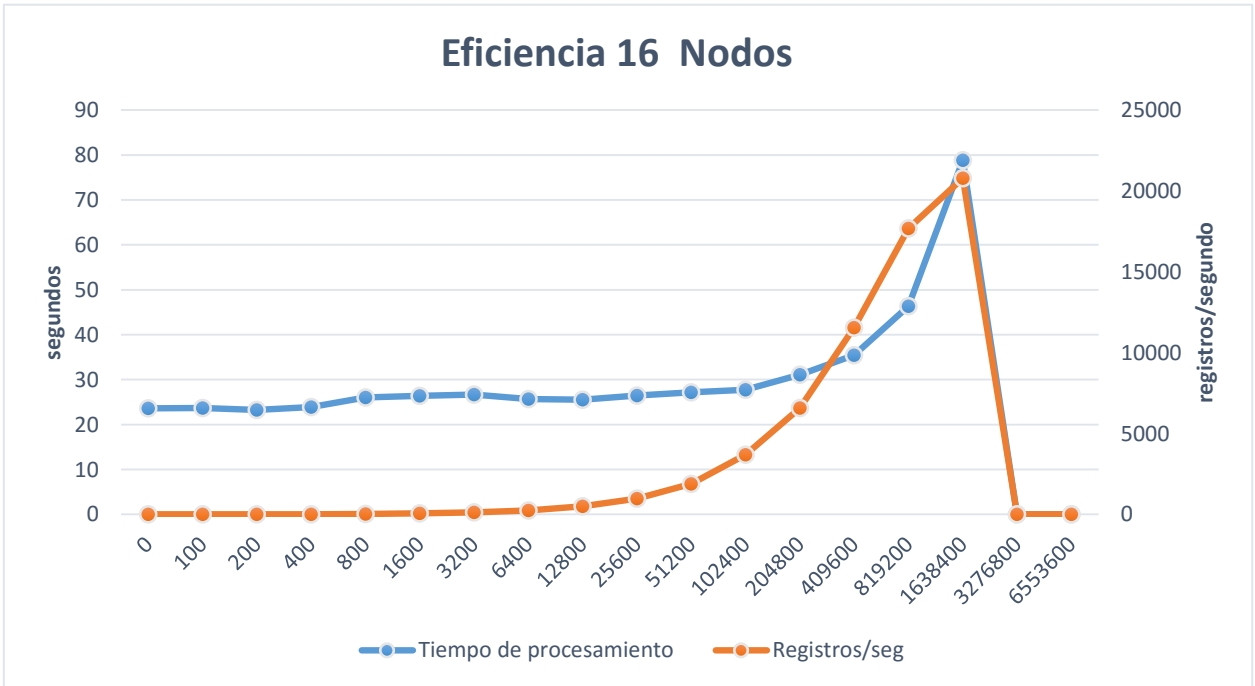


Figura 81: Eficiencia configuración multinodo (16 nodos) – cluster universitario

6.4.5.5. 24 nodos esclavos

Nº de datos	Nombre del	Tiempo de	Registros/Segundo
0	Nasdaq_0.csv	0m24.449s	0
100	Nasdaq_1.csv	0m23.831s	4,19
200	Nasdaq_2.csv	0m25.298s	7,90
400	Nasdaq_3.csv	0m23.689s	16,88
800	Nasdaq_4.csv	0m23.717s	33,73
1600	Nasdaq_5.csv	0m23.801s	67,22
3200	Nasdaq_6.csv	0m25.643s	124,79
6400	Nasdaq_7.csv	0m25.014s	255.85
12800	Nasdaq_8.csv	0m24.866s	514,76
25600	Nasdaq_9.csv	0m25.013s	1023,47
51200	Nasdaq_10.csv	0m26.161s	1957,11
102400	Nasdaq_11.csv	0m28.369s	3609,57
204800	Nasdaq_12.csv	0m30.363s	6745,05
409600	Nasdaq_13.csv	0m35.256s	11617,88
819200	Nasdaq_14.csv	0m47.186s	17361,08
1638400	Nasdaq_15.csv	1m5.012s	25201,50
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0

Tabla 34: Tiempos de procesamiento datos sintéticos cluster universitario (24 nodos)

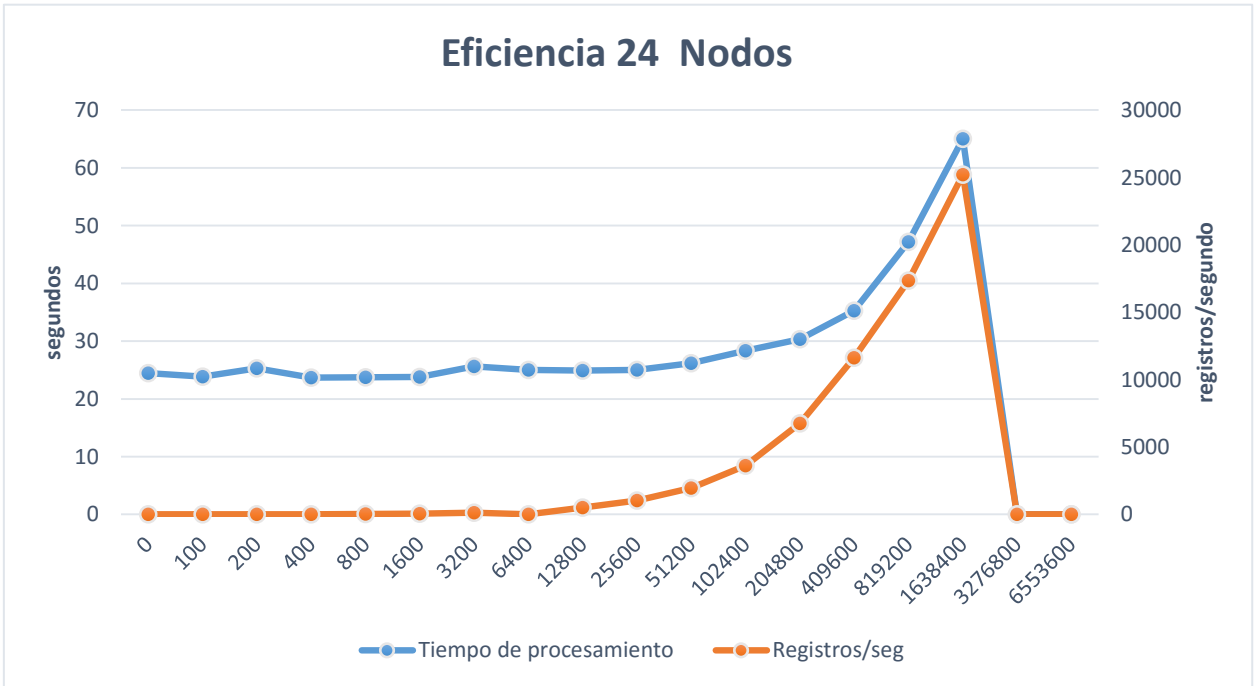


Figura 82: Eficiencia configuración multinodo (24 nodos) – cluster universitario

6.4.6. Comparativa eficiencia arquitectura Hadoop

Para comparar las eficiencias en cada una de las configuraciones, situamos sobre la misma gráfica los resultados obtenidos en los apartados 6.4.3.2 y 6.4.4.2 para tener una visión global de los mismos.

6.4.5.6. Tiempos de procesamiento

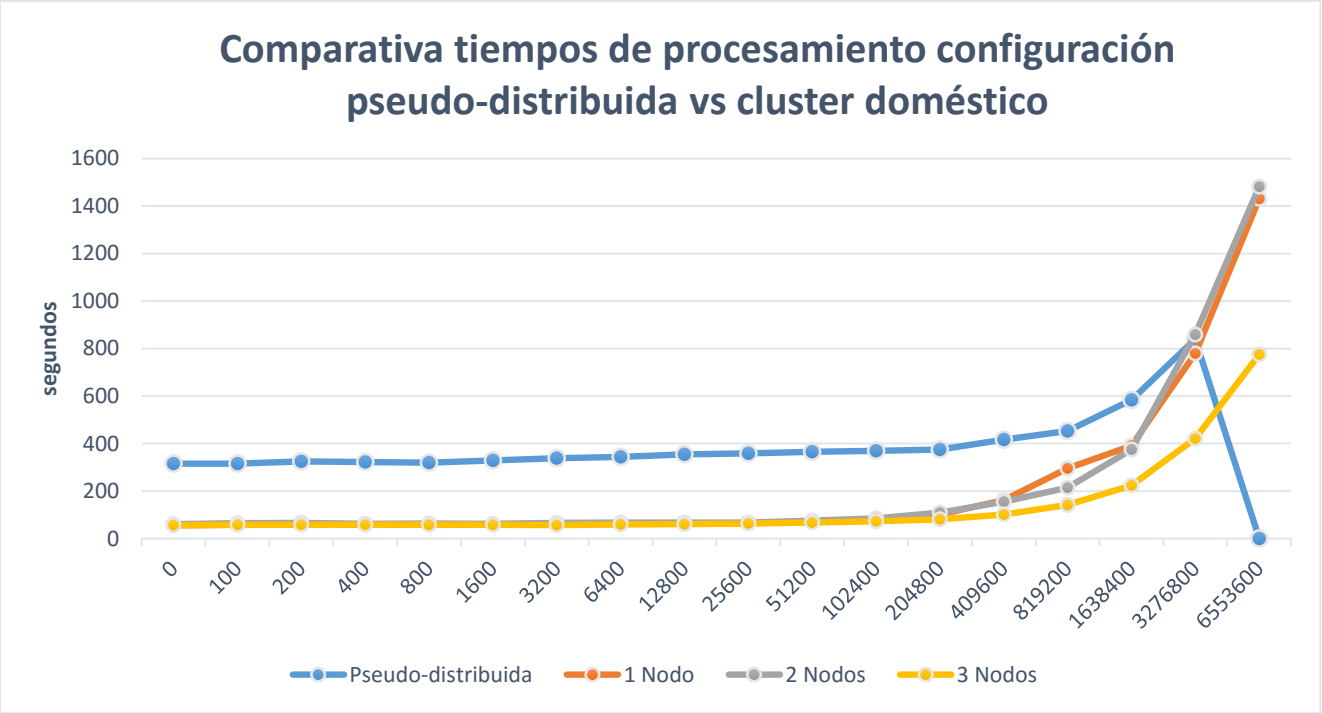


Figura 83: Comparativa tiempos de procesamiento configuración pseudo-distribuida vs cluster doméstico

Observando la Figura 83, la configuración que presenta una mejor eficiencia en cuanto a tiempo de procesamiento es aquella formada por tres nodos esclavos y un maestro, mientras que la peor es aquella que presenta una configuración pseudo-distribuida.

En las cuatro configuraciones encontramos puntos en común:

- Hasta 204800 registros, el tiempo de procesamiento es muy similar, pese a ir aumentando el número de datos. No es hasta que doblamos los datos hasta 409600 cuando el tiempo de procesamiento comienza a crecer. La eficiencia, por tanto, comienza a ser interesante cuando llegamos al punto en el que el tiempo de procesamiento comienza a aumentar de manera significativa, ya que hasta ese número de datos, el coste temporal de procesar los datos es el mismo.

- Una vez se pasa de los 204800 registros, vemos como los tiempos de procesamiento comienzan a aumentar de manera proporcional a como lo hace el número de registros procesados.

6.4.5.7. Número de registros por segundo

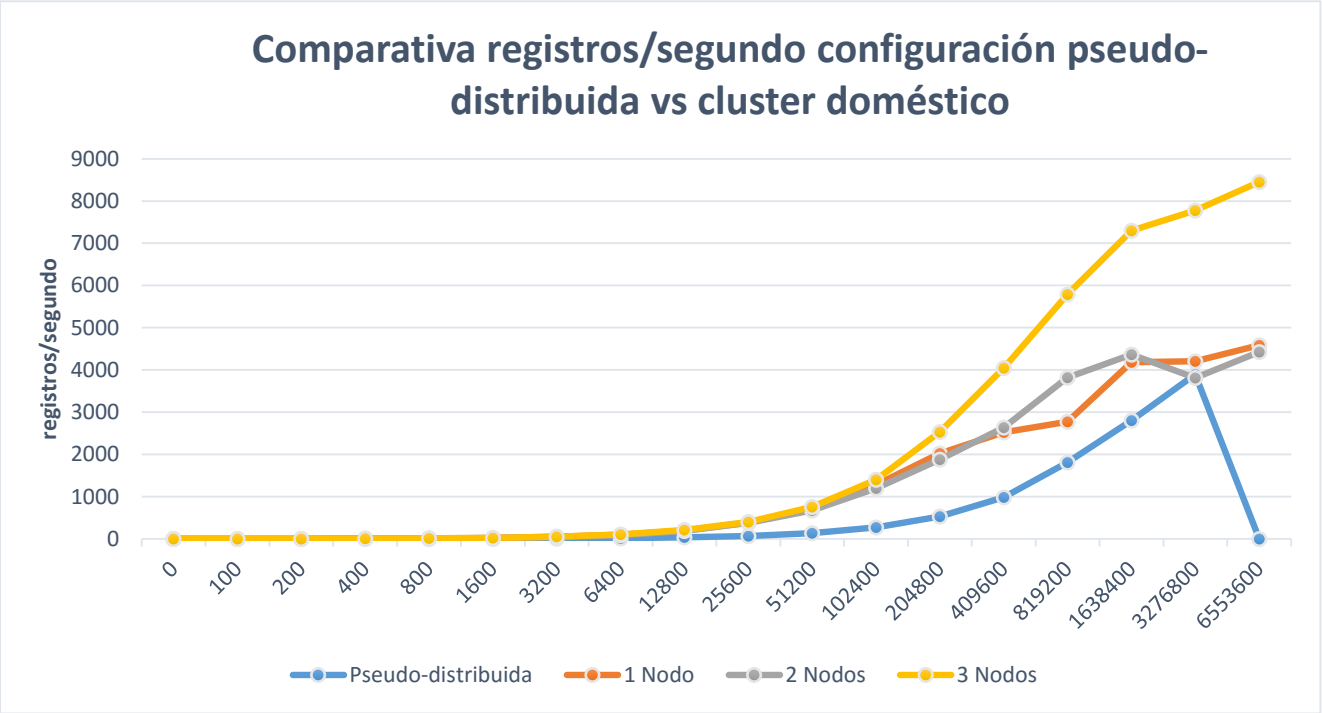


Figura 84: Comparativa registros/segundo configuración pseudo-distribuida vs cluster doméstico

Observando la Figura 84, vemos que el número de registros procesados se mantiene constante hasta los 102400, y a partir de ahí aumenta proporcionalmente a cómo la hace el tiempo de procesamiento y el tamaño en número de datos del fichero.

Cuándo se pasa de 1638400 a 3276800 registros se produce tanto en la configuración multinodo de una máquina como en la de dos máquinas una reducción en las prestaciones que no sucede cuando tenemos tres máquinas. Además en la configuración pseudo-distribuida no se puede finalizar la última ejecución por falta de memoria ram, haciendo que ésta no sea una configuración estable y por tanto recomendable para el procesamiento de datos masivos a partir de 3276800 registros.

Analizando los datos obtenidos se pueden extraer dos interesantes conclusiones de nuestra infraestructura Hadoop:

1. La memoria ram es crítica: Cómo se puede ver en la configuración pseudo-distribuida una memoria ram limitada hace que no podamos procesar un elevado número de registros, dejando el sistema inservible a partir de cierto punto.

2. Número de nodos vs Tiempo de procesamiento, Registros por segundo: Viendo el comportamiento de nuestro sistema con uno y dos nodos, destacamos un empeoramiento en las prestaciones pese a contar con una máquina más y por

tanto más capacidad de computación debido al procesador y la memoria ram de la nueva máquina.

Esto es debido al **factor comunicaciones**, es decir, nos estamos viendo limitados por la propia red en comparación con lo que aporta esa nueva máquina. El envío de datos desde el maestro a los nodos para procesarlos tiene un coste mayor que la capacidad de procesamiento de esa máquina y por lo tanto se produce una reducción en las prestaciones como se puede observar en las figuras (Figura 83, Figura 84).

Por tanto, a la hora de diseñar una arquitectura Hadoop hay que tener muy en cuenta lo expuesto anteriormente. El número de máquinas es tan importante como el número de datos a procesar y las prestaciones de los nodos disponibles en el sistema, teniendo en cuenta el **factor comunicaciones.**

Para contrastar estas conclusiones nos apoyamos en los datos obtenidos después de probar nuestros algoritmos con la configuración multinodo en un cluster universitario dónde el número de máquinas disponibles es más elevado.

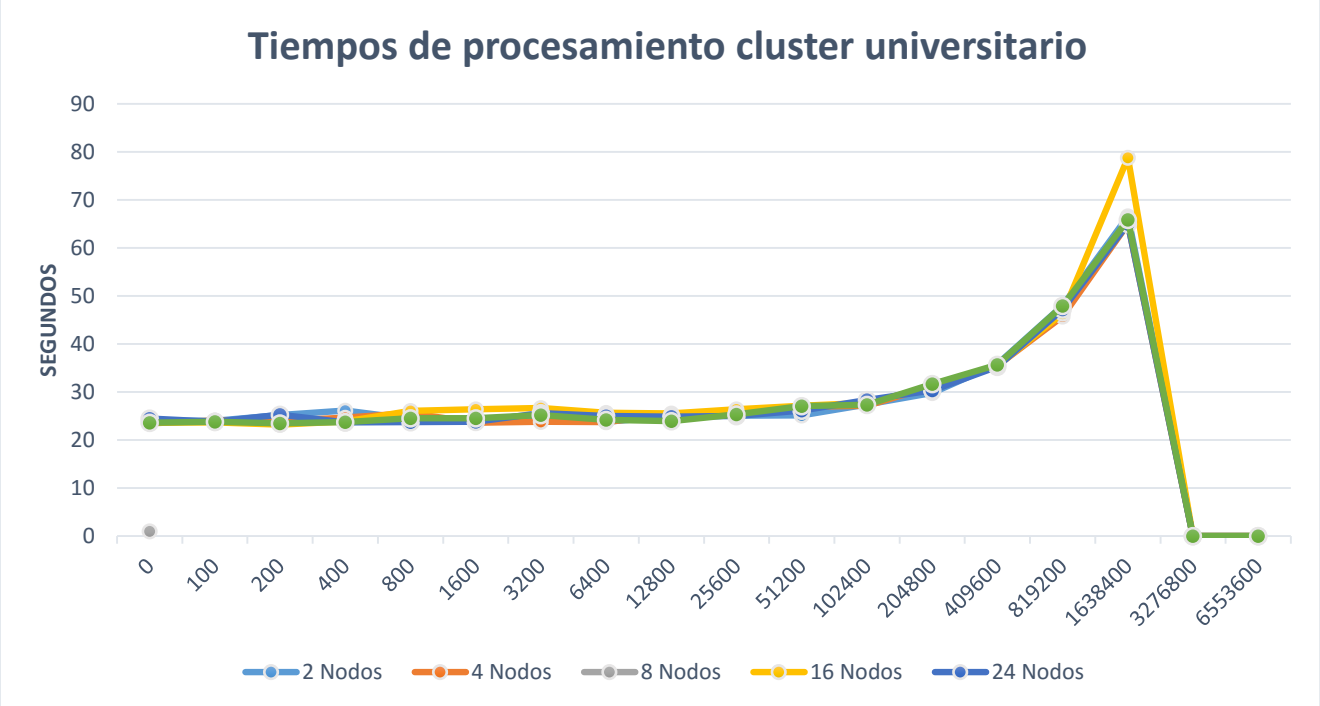


Figura 85: Comparativa tiempos de procesamiento cluster universitario

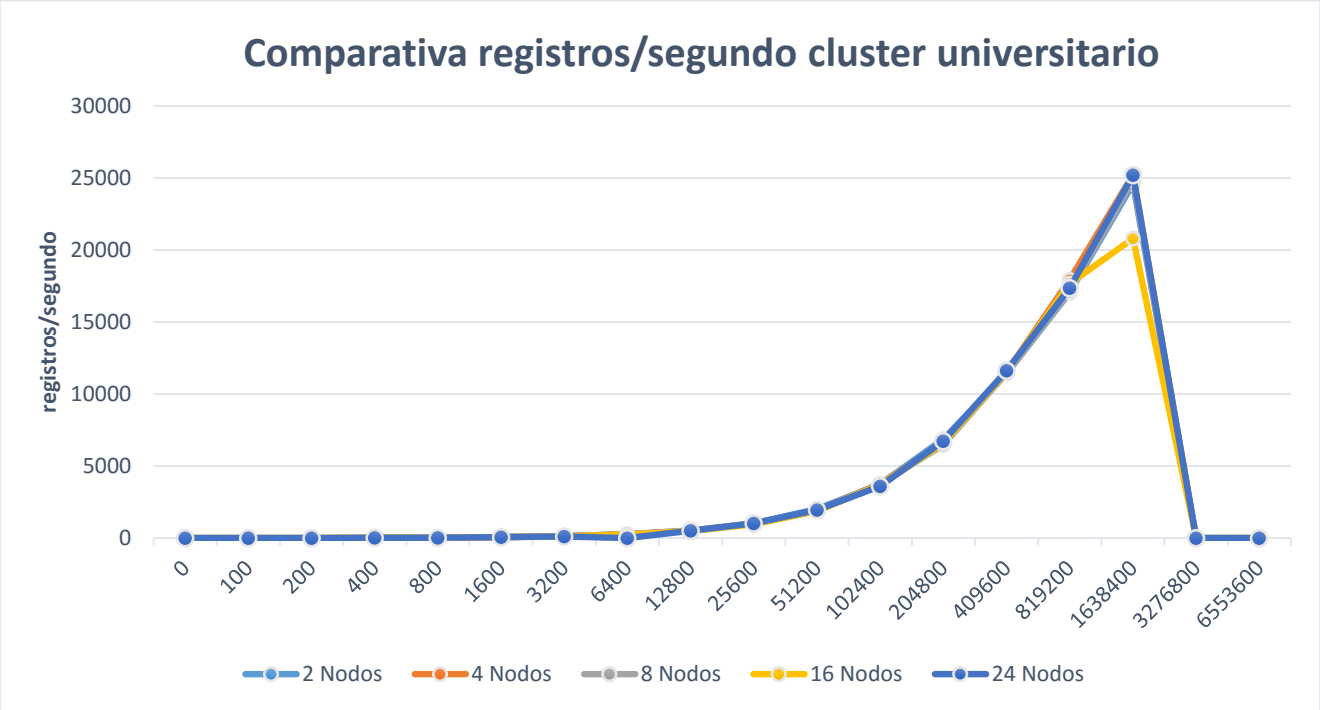


Figura 86: Comparativa registros/segundo cluster universitario

A partir de la Figura 85 y de la Figura 86, se pueden refrendar las conclusiones extraídas. Por un lado vemos en las tablas del apartado 6.4.5 cómo las prestaciones en términos de registros/segundo mejoran de dos a cuatro nodos, empeoran para ocho nodos y se produce un punto característico para dieciséis nodos, dónde la capacidad baja considerablemente. Una vez se supera ese punto crítico, al incorporar 24 nodos esclavos a nuestra infraestructura, mejora de nuevo de forma notable. Este comportamiento viene claramente de la relación número de máquinas/memoria ram/factor comunicaciones explicado anteriormente.

Por otra parte, haciendo una comparativa entre el cluster doméstico y el cluster universitario, vemos cómo el factor comunicaciones afecta incluso de una forma sorprendente al rendimiento del sistema Big Data, tal y cómo vemos en las figuras (Figura 87,Figura 88) de la siguiente página. En ellas se puede ver como en el cluster universitario obtenemos mejores tiempos de procesamiento y por lo tanto una mejora en cuánto al rendimiento de registros por segundo. Tanto el cluster universitario como el doméstico cuentan con el mismo número de nodos esclavos y memoria ram, difiriendo en el tipo de conexiones entre máquinas, fibra óptica frente a cable UTP cat 5e, tal y cómo se especificó en el diseño del sistema en el apartado 3.4.2.

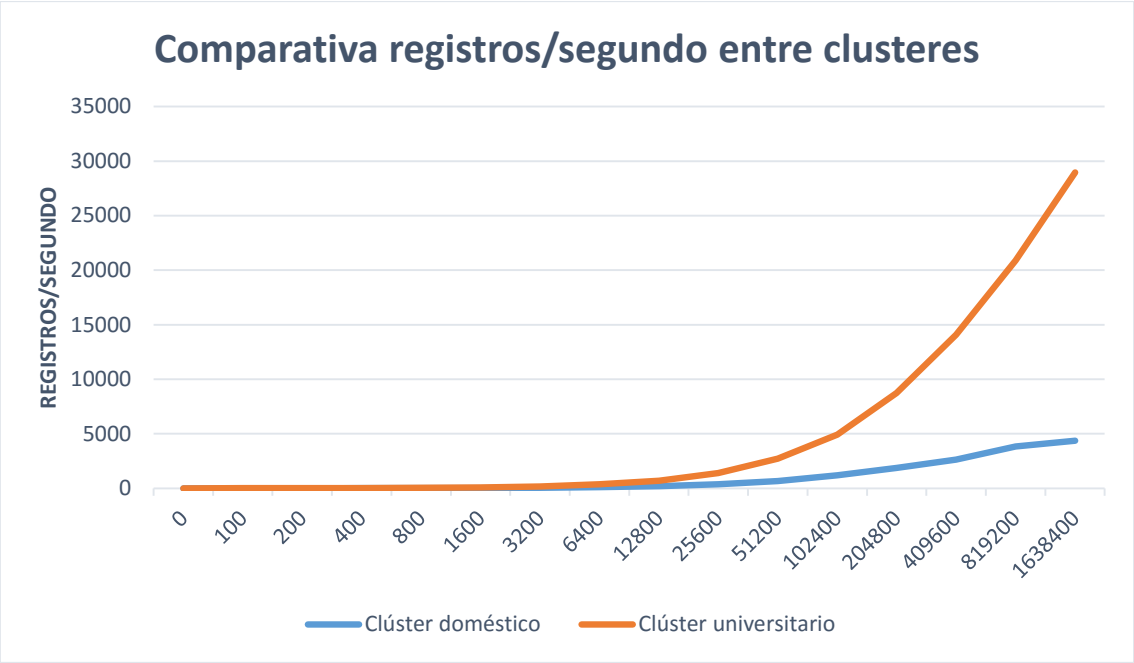


Figura 87: Comparativa registros/segundo entre clusteres

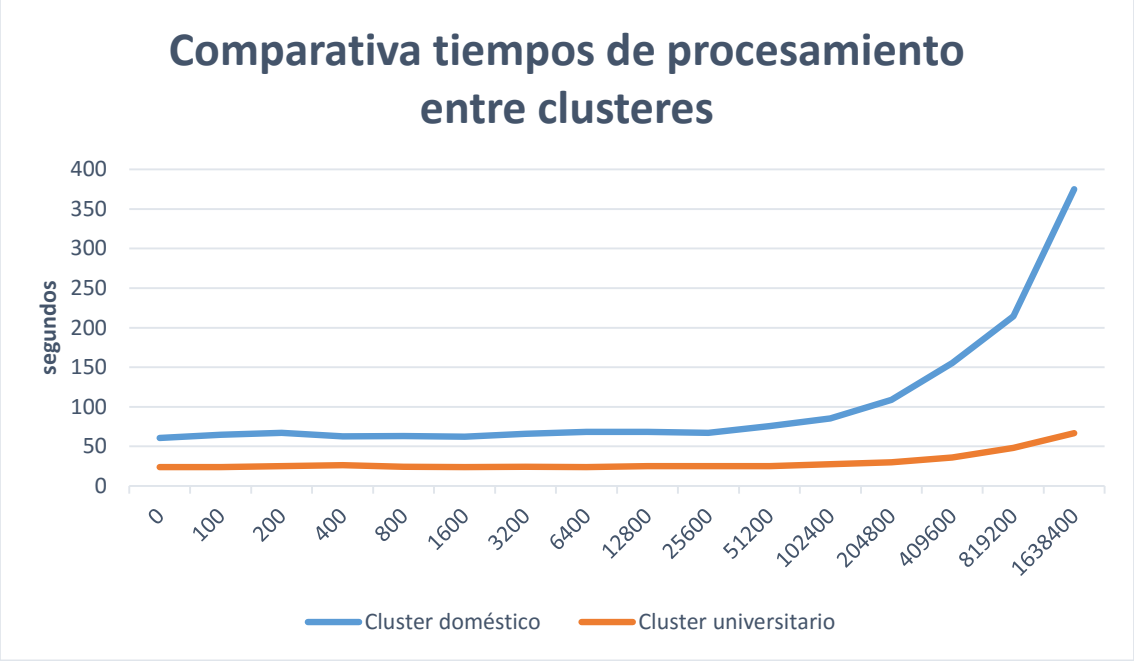


Figura 88: Comparativa tiempos de procesamiento entre clusteres

BLOQUE IV

Conclusiones y líneas futuras de trabajo

7. Conclusiones y líneas futuras de trabajo

7.1. Conclusiones

El objetivo del proyecto era el de diseñar e implementar un sistema Big-data que permitiera el almacenamiento, transformación y análisis de datos procedentes de mercados financieros, contando además con un sistema eficiente y escalable, dotando de una solución polivalente y adaptable a multitud de escenarios y presupuestos.

Después de la realización de las tareas programadas se ha conseguido la consecución de todos los objetivos, contando con una infraestructura eficiente, escalable y polivalente. Además, nos permite el análisis de todo tipo de datos procedentes, en este caso, del ámbito financiero, posibilitando su análisis para obtener los resultados deseados gracias a las tecnologías Hadoop y Hive, resolviendo un análisis de datos muy complejo de una manera sencilla y eficaz.

Además, se ha conseguido detectar los diferentes puntos fuertes y críticos de un sistema de este tipo, mediante diferentes pruebas, detallando los resultados de una forma exhaustiva a partir de tablas y gráficas, posibilitando una posible línea continuista para futuros proyectos

7.2. Futuras líneas de trabajo

Como posibles líneas a seguir en un futuro a partir del trabajo realizado, caben dos:

- Algoritmos económicos: Aumentar el tipo de análisis del mercado financiero a datos diarios para analizar tendencias cortoplacistas en un mundo muy cambiante.
- Arquitectura:
    - Aumento del número de nodos
    - Utilización de un cluster sin limitación del espacio en disco
    - Desarrollo de un cluster con un número de nodos elevado con servicios como AWS [34] de computación en la nube

BLOQUE V

Planificación y presupuesto

8.Planificación y presupuesto

En este apartado se detalla tanto la planificación como el presupuesto del proyecto realizado.

8.1. Planificación

La planificación del proyecto incluye el desglose de todas las tareas, así como su temporización en diferentes fases de desarrollo. La planificación de todas las tareas del proyecto se establece para una duración del mismo de cinco meses para cumplir con el plazo fijado de entrega en la Universidad Carlos III de Madrid, por tanto se desarrolla entre los meses de Febrero y Junio.

8.1.1. Fases de desarrollo

En todas las fases de desarrollo mostradas a continuación participan tanto el alumno como el tutor del TFG, estableciendo un seguimiento a través de reuniones presenciales y métodos electrónicos a lo largo de todo el proyecto con el objetivo de seguir una metodología ágil de trabajo.

1. **Identificación de las necesidades y planteamiento del problema**

   El objetivo de esta fase es el de identificar cual es la necesidad que nos lleva a realizar el proyecto y a partir de ahí plantear el problema y los requisitos, buscando las posibles tecnologías para solucionar ese problema. *Tiempo de ejecución:* 10 días.

2. **Búsqueda y selección de la información**

   A partir de las tecnologías planteadas en la primera fase, se realiza un estudio de las mismas para realizar la implementación de una forma positiva. *Tiempo de ejecución: 15 días.*

3. **Estudio de alternativas de solución**

   En esta tercera fase se estudian las posibles alternativas a partir del problema planteado, fijando y eliminando las no válidas y centrando las alternativas viables y que se acercan de una manera más fiel a los requisitos propuestos. *Tiempo de ejecución*: *10 días*.

4. **Diseño del sistema propuesto**

   Se diseña el sistema propuesto a partir de las alternativas elegidas, especificando todas las posibles configuraciones del mismo con sus elementos y el rol de cada uno de ellos. *Tiempo de ejecución: 20 días*.

5. **Ejecución y configuración del sistema diseñado**

   Una vez ha sido diseñado el sistema, se configuran todos los escenarios elegidos, creando un sistema completamente funcional. *Tiempo de ejecución: 30 días.*

6. **Implementación de las diferentes soluciones propuestas**

   Con el sistema ya creado se procede a la implementación de los algoritmos necesarios para el análisis de los datos procedentes de mercados financieros. *Tiempo de ejecución: 25 días.*

7. **Evaluación del diseño**

   A partir del sistema implementado, incluyendo infraestructura y algoritmos se realiza la evaluación de todo el conjunto, obteniendo resultados con los que obtener conclusiones del trabajo realizado. *Tiempo de ejecución: 20 días*

8. **Redacción de la memoria**

   Documentación de todo el trabajo realizado durante las siete fases anteriores. *Tiempo de ejecución: Desarrollo continuo desde la fase 2. Finalización de la misma: 10 días*

Fases críticas del proyecto

A lo largo del desarrollo del proyecto encontramos dos fases críticas, en caso de problemas en las mismas se retrasará la ejecución de las fases siete y ocho, no afectando al cómputo global al trabajar durante la planificación con márgenes de ejecución de cara a evitar retrasos en la entrega final del mismo. Las dos fases críticas son las siguientes:

- Fase 5 (Ejecución y configuración del sistema diseñado): En caso de producirse retrasos en la configuración del sistema no se podrá evaluar el diseño a partir de los algoritmos creados en la fase 6.
- Fase 6 (Implementación de las diferentes soluciones propuestas): Al igual que en la fase 5, el retraso en esta fase no permitiría la evaluación global del sistema.

8.1.2. Diagrama de Gantt

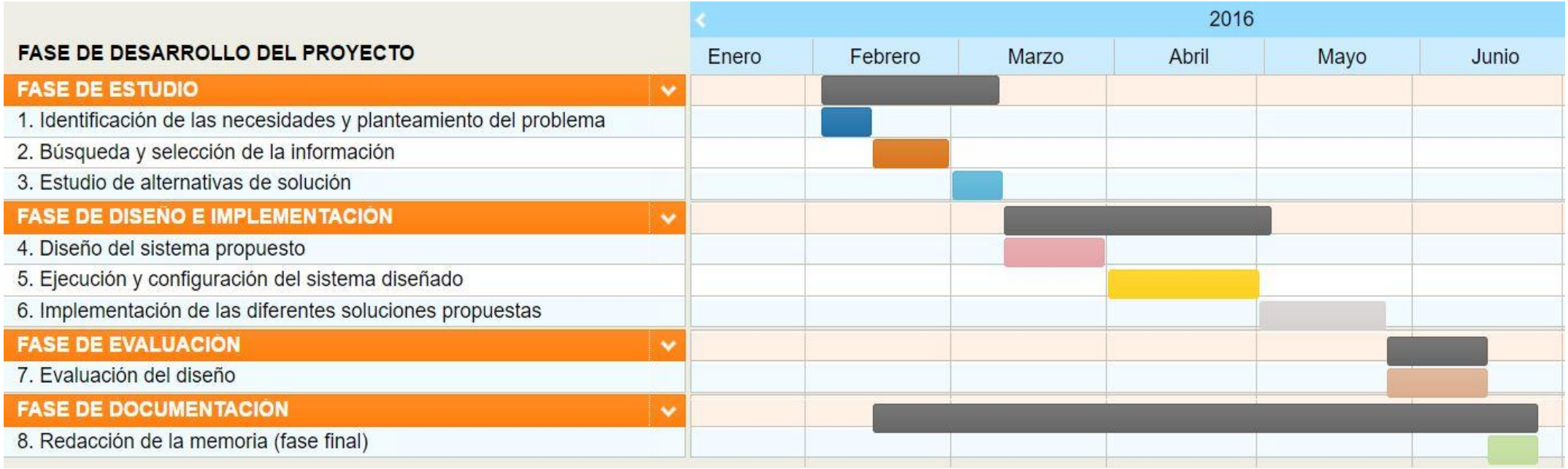


Figura 89: Diagrama de Gantt - fases de desarrollo del proyecto

8.2. Presupuesto

En este apartado se van a detallar los costes de realización de este proyecto de fin de grado, incluyendo los medios empleados, para finalmente, mostrar un presupuesto del mismo.

8.2.1. Medios empleados

Los medios empleados pueden ser de tres tipos según su naturaleza:

- Medios personales: A lo largo del desarrollo del proyecto participan dos personas, Pablo Basanta Val, reflejado como ingeniero senior, y Miguel Aller Camino, como ingeniero junior.
- Medios materiales: En la realización del proyecto se emplean una serie de recursos materiales para llevar a cabo los escenarios diseñados:
    - 1 ordenador portátil Toshiba C40D-A
    - 1 ordenador portátil Hp Pavillion
    - 2 ordenadores sobremesa Hp Pavillion
    - 1 memoria usb verbatim 32 Gb
- Medios inmateriales: Para ejecutar la implementación del proyecto se emplean una serie de programas, tanto con licencia, como open-source, que se exponen a continuación:
    - 1 Microsoft Office 2013 Universidad
    - 4 Ubuntu 15.10
    - 1 Apache Hadoop 2.7.2
    - 1 Apache Hive 1.2.1
- Otros Medios:
    - 2 Aulas telemática Uc3m (4 horas)
    - 3 Conexión monitores aulas telemática Uc3m (4 horas)
    - Conexión a internet (6 meses)

8.2.2. Presupuesto

El presupuesto se realiza teniendo en cuenta los recursos especificados en el apartado anterior (8.2.1). Se incluyen todos los elementos, aquellos con algún coste y también los que tienen coste cero. Se divide por tipo de medio para después ofrecer un presupuesto total del proyecto.

Medios personales

Participan dos personas, cada una con un rango diferente en el entorno laboral, por lo que el coste imputable es diferente en cada caso, para el cálculo de este apartado se han tenido en cuenta los siguientes sueldos:

-Sueldo Ingeniero Junior/hora: 15 euros

-Sueldo Ingeniero Senior/hora: 25 euros

Para conocer el coste imputable a cada persona del proyecto hacemos un desglose de las actividades realizadas y las horas que supone cada una de ellas:

Fase de desarrollo	Días	Horas Ingeniero Junior	Horas Ingeniero Senior
Identificación de las necesidades y planteamiento del problema	10	40	4
Búsqueda y selección de la información	15	60	5
Estudio de alternativas de solución	10	40	10
Diseño del sistema propuesto	20	80	10
Ejecución y configuración del sistema diseñado	30	120	15
Implementación de las diferentes soluciones propuestas	25	100	20
Evaluación del diseño	20	80	15
Redacción de la memoria	10	40	10
TOTAL	140	560	89

Tabla 35: Detalle de horas empleadas por fases de desarrollo

Una vez han sido analizadas el número de horas empleadas por cada uno de los participantes calculamos el coste que ello supone:

COSTE MEDIOS PERSONALES				
Concepto	**Cantidad**	**Detalle**	**Coste Hora (euros)**	**Coste Total (euros)**
Ingeniero Junior	1	560 Horas	15	8400
Ingeniero Senior	1	89 Horas	25	2225
TOTAL				10625

Tabla 36: Costes de los medios personales

Medios materiales

COSTE MEDIOS MATERIALES					
Concepto	**Cantidad**	**Precio (euros)**	**Depreciación(Coeficiente máximo anual)[3]**	**Periodo de Uso (meses)**	**Coste Total imputable (euros)**
Ordenador portátil Toshiba C40D-A	1	560	25%	6	70
Ordenador portátil Hp Pavillion	1	610	25%	6	76,25
Ordenador sobremesa Hp	2	670	25%	6	83,75
Memoria usb verbatim 32 Gb	1	20	-	6	20
TOTAL					250

Tabla 37: Costes de los medios materiales

Medios inmateriales

COSTE MEDIOS INMATERIALES					
Concepto	**Cantidad**	**Precio (euros)**	**Amortización**	**Periodo de Uso (meses)**	**Coste Total imputable (euros)**
Microsoft Office 2013 Universidad	1	0	33%	6	0
Ubuntu 15.10	4	0	33%	6	0
Apache Hadoop 2.7.2	1	0	33%	6	0
Apache Hive 1.2.1	1	0	33%	6	0
TOTAL					0

[3] Coeficiente lineal máximo de amortización fijado en el BOE del 28 de noviembre de 2014 [35]

Tabla 38: Costes de los medios inmateriales

Otros medios

COSTE OTROS MEDIOS				
Concepto	**Cantidad**	**Detalle**	**Coste**	**Coste Total (euros)**
Alquiler aula telemática Uc3m	2	4 horas	0	0
Conexión monitores aula	3	4 horas	0	0
Conexión a internet	1	6 meses	20 euros/mes	120
TOTAL				120

Tabla 39: Costes de otros medios

Una vez han sido estudiados todos los costes derivados de la realización del proyecto, divididos por tipos de medio, exponemos el coste total final:

COSTE TOTAL DEL PROYECTO	
Concepto	**Coste (euros)**
Medios personales	10625
Medios materiales	250
Medios inmateriales	0
Otros medios	120
TOTAL	10995

Tabla 40: Coste total del proyecto

Por tanto, el coste total del proyecto, ejecutado en un plazo de 6 meses queda fijado en 10995 euros.

Bibliografía

[1] *Yahoo Finance*. (Marzo de 2016). Obtenido de http://finance.yahoo.com/

[2] *Apache Hadoop*. (Febrero de 2016). Obtenido de http://hadoop.apache.org/

[3] *Apache Hive*. (Febrero de 2016). Obtenido de https://hive.apache.org/

[4] *IBM*. (Febrero de 2016). Obtenido de https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data/

[5] *Forbes*. (Marzo de 2016). Obtenido de http://www.forbes.com/sites/louiscolumbus/2014/06/24/roundup-of-analytics-big-data-business-intelligence-forecasts-and-market-estimates-2014/#5d2d86c45466

[6] *Data versity*. (Marzo de 2016). Obtenido de http://www.dataversity.net/wp-content/uploads/2012/06/Sunil-Blog-1.png

[7] *Microsoft*. (marzo de 2016). Obtenido de https://msdn.microsoft.com/es-es/library/ms174949.aspx

[8] *Computación distribuida*. (marzo de 2016). Obtenido de http://3.bp.blogspot.com/_y0PxP8AApJ0/TK-o1wG_42I/AAAAAAAAAWw/qWx2FGiC0WQ/s1600/computaciondistribuida.jpg

[9] Granada, U. d. (Febrero de 2016). Obtenido de http://es.slideshare.net/JacF/cluster-beowulf-javier-condori-flores?qid=425e16f6-9e87-436a-b2f0-fbd75d49e882&v=&b=&from_search=7

[10] *Cluster*. (marzo de 2016). Obtenido de http://clusterfie.epn.edu.ec/clusters/Definiciones/Images/componentes.gif

[11] *Daemons*. (marzo de 2016). Obtenido de https://media.licdn.com/mpr/mpr/shrinknp_800_800/AAEAAQAAAAAAAAVaAAAAJGEzNzlmNTFlLWE3OGUtNDQzNS05OWViLWEwZDVhMWRiNWRiMA.jpg

[12] *Hds Apache Hadoop*. (marzo de 2016). Obtenido de https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[13] *Mapreduce*. (marzo de 2016). Obtenido de http://alumni.cs.ucr.edu/~jdou/misco/figs/mapreduce.png

[14] ISO. (s.f.). *Big Data, preliminary report 2014.* Obtenido de http://www.iso.org/iso/big_data_report-jtc1.pdf

[15] España, G. d. (s.f.). *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.* Obtenido de https://www.boe.es/diario_boe/txt.php?id=BOE-A-1999-23750

[16] *Agencia Española de Protección de datos*. (Marzo de 2016). Obtenido de http://www.agpd.es/portalwebAGPD/internacional/Europa/grupo_29_europeo/index-ides-idphp.php

[17] Europea, C. (s.f.). Obtenido de http://ec.europa.eu/justice/data-protection/article-29/documentation/opinion-recommendation/files/2014/wp216_en.pdf

[18] *Ubuntu*. (Febrero de 2016). Obtenido de http://www.ubuntu.com/

[19] White, T. (2015). *Hadoop: The Definitive Guide, 4th Edition.* O'Reilly Media

[20] *Ubuntu releases*. (Febrero de 2016). Obtenido de http://releases.ubuntu.com/

[21] *Rufus*. (Febrero de 2016). Obtenido de https://rufus.akeo.ie/

[22] *Requisitos previos hadoop*. (Febrero de 2016). Obtenido de http://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-common/SingleCluster.html#Pseudo-Distributed_Operation

[23] *Pingax*. (Marzo de 2016). Obtenido de http://pingax.com/install-apache-hadoop-ubuntu-cluster-setup/

[24] *webupd8*. (Febrero de 2016). Obtenido de http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-via-ppa.html

[25] *Apache releases*. (Febrero de 2016). Obtenido de http://hadoop.apache.org/releases.html

[26] Edward Capriolo, D. W. (2012). *Programming Hive.* O'Reilly Media.

[27] *Apache Hive Releases*. (Marzo de 2016). Obtenido de http://apache.rediris.es/hive/stable/

[28] *Mapr*. (Abril de 2016). Obtenido de https://www.mapr.com/blog/quick-tips-using-hive-shell-inside-scripts

[29] Ugarte, J. I. (1999). *El análisis técnico bursátil.*

[30] Ajram, J. (2013). *Bolsa Para Dummies.*

[31] Alba, U. N. (2012). *Efecto mariposa y crisis financiera - Fallos regulatorios.*

[32] *Getting Started With Hadoop*. (Abril de 2016). Obtenido de https://wiki.apache.org/hadoop/GettingStartedWithHadoop

[33] Ynoub, G. (s.f.). *Japón: de la economía burbuja a la recesión.* Obtenido de http://nulan.mdp.edu.ar/602/1/ynoub_g.pdf

[34] *Amazon AWS*. (s.f.). Obtenido de https://aws.amazon.com/es/

[35] *BOE.* (mayo de 2016). Obtenido de https://www.boe.es/boe/dias/2014/11/28/pdfs/BOE-A-2014-12328.pdf

Anexo I: Summary

1. Introduction and objectives

1.1. Context and motivation

In recent years, the traffic and generation of data has raised significantly as a result of the increase in the number of devices available on the market and also in the amount of people with access to the internet network. The Smartphone and tablet Era, and most recently the wearables and The Internet of Things age, have caused the existing view on the analysis of current data to become old-fashioned and fall behind the times, due to the impossibility to absorb the high and growing demand derived from these new gadgets and functionalities.

The companies and different organizations look to obtain a competitive advantage through the smart use of the enormous amount of data available to define new business strategies that would allow them to increase their revenues or, in the case of governments and other social functions, allow them to achieve a good and efficient functioning.

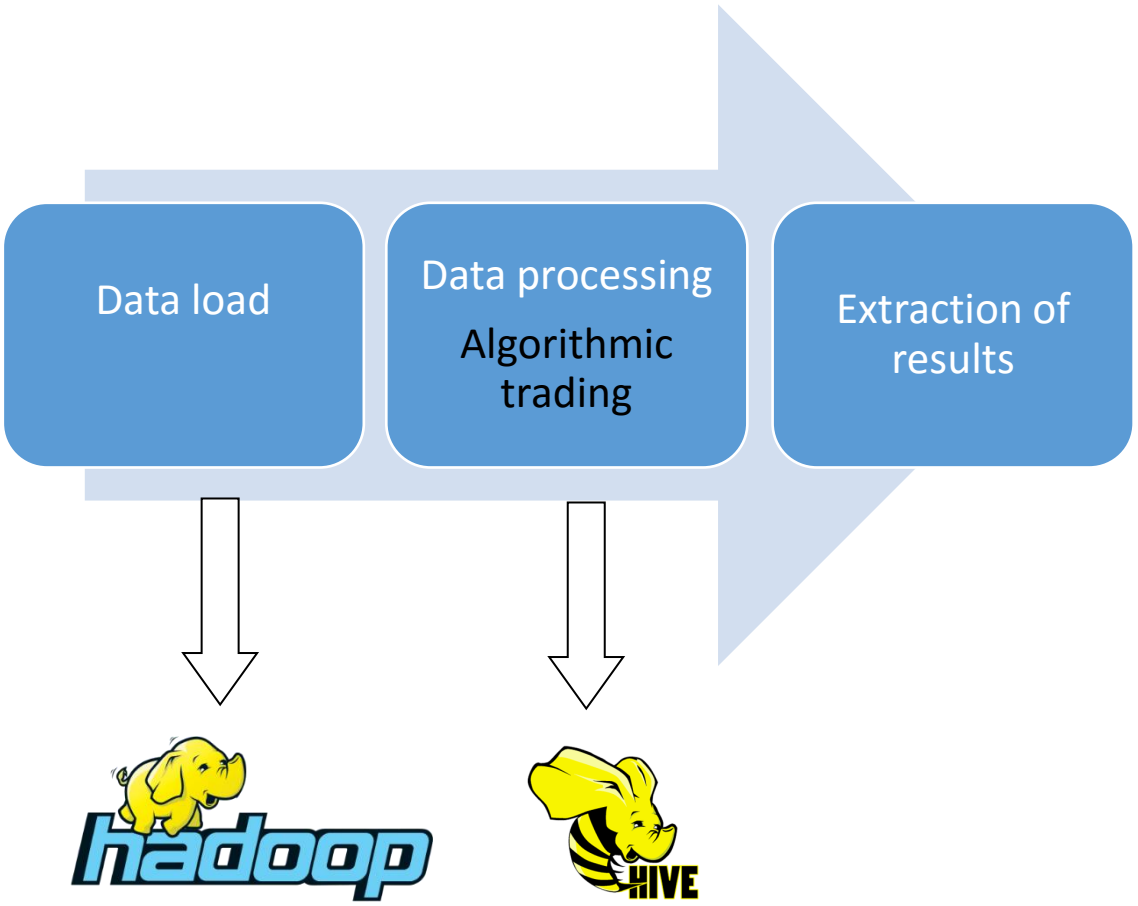
When talking about financial markets, the importance of the data is simply key. Investment funds, private banking, commercial banks, and also governments, need efficient, fast and powerful tools to analyse data from which true and immediate results can be obtained. Nowadays, this is getting more and more difficult as the traditional tools require an excessive computational cost.

Taking into account these necessities, this project focuses on the need for the financial services sector and the global stock markets to find efficient tools for analysis of Big-Data. As part of our study, we will analyse historic trends to generate profitable algorithms to analyse and forecast financial markets (algorithmic trading) and obtain high yields and returns on our investments. And the reality is that the computerized algorithms have already started to substitute the stock analysts in Wall Street.

From free accessible data, we interact with a Big Data system based on two technologies: Apache Hadoop y Apache Hive in order to process and analyse the collected data and extract valuable conclusions. In this context, once we have obtained the data, we load it into the system, which processes it through generated algorithms (based on Hive commands), we analyse it and extract relevant conclusions from the results obtained.

The Big Data system allows a wide range of potential solutions that will be analysed throughout our study. Each of the various solutions presents a different scenario with different efficiency levels for the generated infrastructure.

Therefore, the overview of the main steps and respective functionalities of the Big Data system object of our study are as follows:



1.2. Objectives

The main objective of our project is the design and implementation of an efficient, scalable and versatile Big Data system based on the Hadoop and Hive technologies for the analysis of the financial markets.

In order to reach this goal, the project requires the following steps:

- The analysis and study of the different technologies required for the implementation of the system (Apache Hadoop, Apache Hive).
- Knowledge on the different needs of the financial markets.
- The design of a Big Data system with different configurations.

- The implementation of the designed Big Data system in different environments.
- The extraction of results from the different algorithms generated to test the financial markets and produce reliable forecasting (algorithmic trading).
- Learning how to use the designed system as well as the scripts for further implementation.
- The test and analysis of the efficiency of the system on the different configurations and environment (scenarios).
- The extraction of conclusions from the results obtained on our system implementation testing exercise and the definition of future steps and work lines.
- The preparation of a detailed budget that would determine the economic size of the project.

2. Tests and results

2.1. Introduction

Once we have finalized the implementation of our new Big-data architecture, we will proceed to carry out certain tests over different scenarios with the aim of finding any limitation in the system as well as getting to know its scalability and explore its bounders and possibilities. Furthermore, we will study the economic efficiency of the different algorithms we have generated related to financial investments.

In order to develop these tests, we will test two types of configurations, pseudo-distributed and multinode. In both cases, we will use original data files directly from the financial markets as well as synthetic data to test the performance of the infrastructure on a scenario where a high load of data is required.

2.2. Environment of the test

For each type of configuration, there is a different number of computers, each of them with specific characteristics that will be explained on the following sections:

2.2.1. Pseudo-distributed configuration

In this case there is only one computer that works as master and slave. The configuration is as follows:

General
Procesador: 4x AMD A4-5000 APU with Radeon(TM) HD Graphics
Almacenamiento: 5543MB (2215MB used)
Sistema Operativo: Ubuntu 15.10
Usuario: miguelaller (MiguelAller)
Procesadores
AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz
AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz
AMD A4-5000 APU with Radeon(TM) HD Graphics 800,00MHz
AMD A4-5000 APU with Radeon(TM) HD Graphics 1300,00MHz
Almacenamiento
ATA TOSHIBA MQ01ABF0
Red
Controlador Ethernet: Qualcomm Atheros AR8162 Fast Ethernet
Controlador de red: Realtek Semiconductor Co., Ltd. RTL8188EE Wireless Network Adapter

2.2.2. Domestic cluster

In this case, however, we count with different combinations:

- 1 node: one master and one slave
- 2 nodes: one master and two slaves
- 3 nodes: one master and three slaves

In total, we are using four computers, each of them with its distinct configuration:

Master: Same configuration as pseudo-distributed master

Slave 1:

General
Procesador: 4x Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz
Memoria ram: 4046MB
Sistema Operativo: Ubuntu 15.10
Usuario: miguelaller (MiguelAller)
Proccesors
Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2000,00MHz
Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2000,00MHz
Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2333,00MHz
Intel (R) Core (TM) 2 Quad CPU Q8200 @ 2.33GHz 2333,00MHz
Storage
ATA WDC WD10EADS-65L

Network
Controlador Ethernet: Realtek Semiconductor. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller Controlador de red: Ralink corp. RT2790 Wireless 802.11n 1T/2R PCIe

Slave 2:

General
Procesador: 4x Intel(R) Core(TM) i5 CPU M 460 @ 2.53GHz Memoria ram: 3841MB Sistema Operativo: Ubuntu 15.10 Usuario: miguelaller (MiguelAller)
Proccesors
Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2534,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 1733,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2534,00MHz Intel (R) Core (TM) i5 CPU M460 @ 2.53GHz 2399,00MHz
Storage
ATA SAMSUNG HM641JI
Network
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller Controlador de red: Ralink corp. RT2790 Wireless 802.11n 1T/2R PCIe

Slave 3:

General
Procesador: 4x Intel(R) Core(TM) i3 CPU M 460 @ 1.83GHz Memoria ram: 4386 MB Sistema Operativo: Ubuntu 15.10 Usuario: hduser ()
Proccesors
Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1500,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1500,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1630,00MHz Intel (R) Core (TM) i3 CPU M460 @ 1.83GHz 1630,00MHz
Storage
ATA WDC WD5000AAKS-6

Network
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL8101/2/6E PCI Express Fast/Gigabit Ethernet controller Controlador de red: Ralink corp. RT3092 Wireless 802.11n 2T/2R PCIe

As we can see in the tables below, masters and slaves are quad core computers and with 4 GB of ram memory, warranting the homogeneity of the configuration, but not an ideal system. All the computers contribute with the same resources.

8.1.2. Universitary Cluster

In this configuration, all the computers contribute with the same resources, doing an ideal cluster. All the computers have this configuration:

General
Procesador: 4x Intel(R) Core(TM) i3-3240 CPU @ 3.40GHz Memoria Ram: 3310MB Sistema Operativo: Debian GNU/Linux 8.1 Usuario: 0284736 (GRADO EN INGENIERIA DE SISTEMAS DE COMUNICACIONES)
Proccesors
Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz Intel (R) Core(TM) i3-3240 CPU @ 3.40GHz 3400,00MHz
Storage
ATA WDC WD5000AAKX-0 TSSTcorp CDDVDW SH-224DB
Network
Controlador Ethernet: Realtek Semiconductor Co., Ltd. RTL-8100/8101L/8139 PCI Fast Ethernet

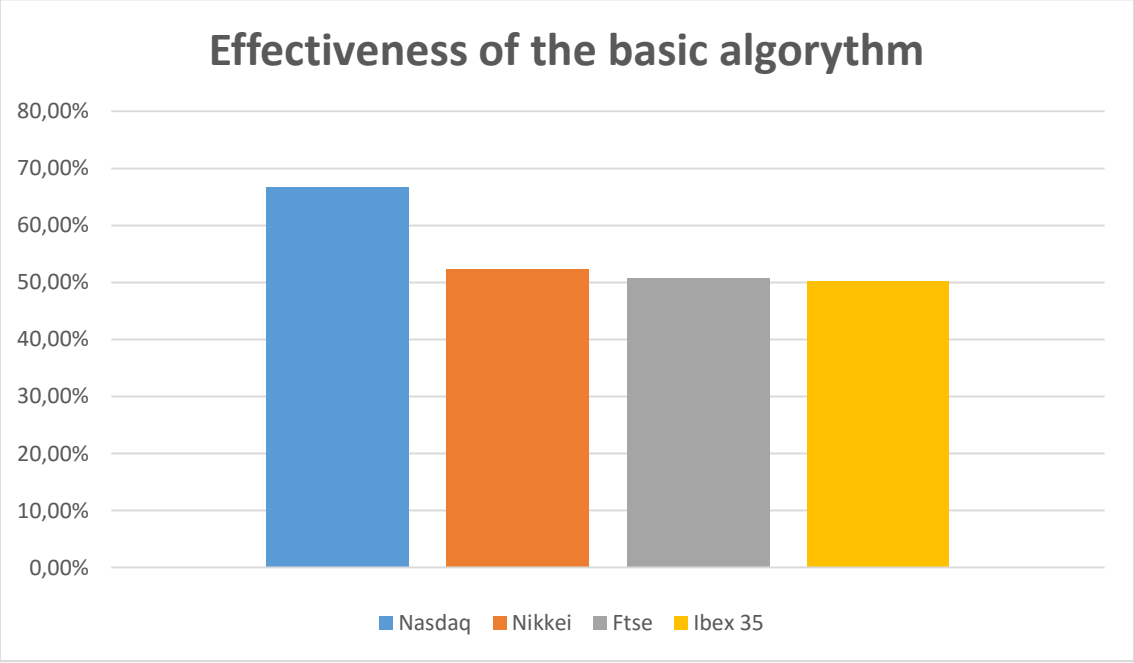
2.3. Effectiveness of the generated algorithms

2.3.1. Effectiveness of the basic algorithm

The final goal of the algorithm to be implemented is to create a tool that would allow trustable forecast. This is the reason why the success rate is, in this case, decisive.

On the chart below, we find the results obtained in terms of success and failure on our analysis of the different stock exchanges in scope of our project.

Algorithm	Stock Exchange	Data population	Number of successes	Number of failures	% of success (success rate)
algoritmo_base_simplificado.hql	Nasdaq	11388	7603	3784	66.77 %
algoritmo_base_simplificado.hql	Ftse	8408	4269	4138	50.78 %
algoritmo_base_simplificado.hql	Nikkei	7949	4162	3786	52.37 %
algoritmo_base_simplificado.hql	Ibex 35	5813	2918	2894	50.21 %



As shown on the table and chart above, the market that presents the best results is the "Nasdaq". In this market we count with the biggest quantity of data. By contrast, the market with the worst results is the "Ibex 35", for which we have a smaller amount of data.

As we have analysed a small sample of markets, we cannot extrapolate the results and conclude on a general rule; however, we can say that the bigger the amount of data we have, the best we can predict the future behaviour of the financial markets thanks to common trends observed on the historical behaviour of the Stock Exchanges over certain periods of time in the past.

2.3.2. Effectiveness of the advanced algorithm

On the table below, we have included the results obtained on our analysis of the "Ibex 35" when combining data from different markets:

Algorithm	Stock Exchange	Amount of processed data	Number of successes	Number of failures	% of success
algoritmo_avanzado.hql	Ibex Nasdaq Nikkei	23252	2893	2920	49.77 %

By contrast of what we could have expected when combining data from different Stock Exchanges, the effectiveness obtained is lower than the one achieved when applying the basic algorithm.

However, if we take the Japanese Stock Exchange (Nikkei) as an example, we observe that the fact that the Japanese economy is in continuous recession and as a consequence the financial market is in a downtrend (bearish market), directly impacts our prediction exercise bringing errors to our analysis executed on the "Ibex 35".

2.4. Efficiency of the infrastructure

2.4.1. Introduction

Once we have obtained the results in terms of performance of the algorithms we have generated and therefore tested their effectiveness, we are going to study now the performance (in terms of efficiency) of our Hadoop infrastructure by feeding data into the system through the different algorithms generated (basic and advanced).

The data we have used for our efficiency test is the result of the combination of the data together with synthetic data.

The performance of the infrastructure will be tested in two different configurations: pseudo-distributed and multinode. For the results and conclusions of our test, refer to the sections below.

2.4.2. Synthetic data

The objective of this part of our project is to study the amount of time required by the infrastructure to analyse synthetic data, and not to test the results of the algorithm generated, as in the sections above.

In this context, as we do not have enough data from the sources used in our study, we are going to create synthetic data with the goal of analysing the boundaries and limits of the infrastructure. For this, we will model the available data using exponential functions (multiplying then by two the amount of data on each iteration) in order to build a population large enough to analyse the relationship between the amount of data and the time of data processing.

For this exercise, we have chosen the "Nasdaq" as it is the market for which we have collected the biggest amount of data. When multiplying the existing data by two, we obtain the following data population per iteration:

Data population (line items)	Size of the data file	Name of the file
0	41B	Nasdaq_0.csv
100	8,1 kB	Nasdaq_1.csv
200	16,2 kB	Nasdaq_2.csv
400	32,4 kB	Nasdaq_3.csv
800	64,6 kB	Nasdaq_4.csv
1600	128,9 kB	Nasdaq_5.csv
3200	257,4 kB	Nasdaq_6.csv
6400	506,3 kB	Nasdaq_7.csv
12800	954,4 kB	Nasdaq_8.csv
25600	1,9 MB	Nasdaq_9.csv
51200	3,8 MB	Nasdaq_10.csv
102400	7,6 MB	Nasdaq_11.csv
204800	15,3 MB	Nasdaq_12.csv
409600	30,5 MB	Nasdaq_13.csv
819200	61,1 MB	Nasdaq_14.csv
1638400	123,8 MB	Nasdaq_15.csv
3276800	244,3 MB	Nasdaq_16.csv
6553600	488,6 MB	Nasdaq_17.csv

The first value on the table above is 0 (no data), in order to test the processing time required by the system "Hive" to run the algorithm with no data. As a result, we are able to identify how much of the processing time is required by the architecture itself and how much is used for the data processing. In this case, even though there is no data

being processed, the size of the data file is not zero as it contains the header information.

The process for creating synthetic data is very simple; we apply the exponential function to the existing data, duplicating the amount of data on the original file, and therefore obtain on each step a data file with double the data of the one on the previous file.

For the analysis of the new files generated synthetically, we will apply a simplified algorithm (basic) that requires a lower execution time between one file and the next one, as only two changes are required: the variable “length” and the source file from which the data is loaded. Please see below the aforementioned algorithm:

algoritmo_base_simplificado_datos_sintéticos.hql

```
set longitud=(longitud del fichero a analizar);

create database IF NOT EXISTS basedatos_financiera_sinteticos;
use basedatos_financiera_sinteticos;

create table IF NOT EXISTS mercado_sinteticos (fecha string,open float,high
float,low float,close float,volume float,adjclose float) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
load data local inpath '/home/miguelaller/Documentos/TFG/Datos/(mercado a
analizar).csv' overwrite into table mercado_sinteticos;

drop table IF EXISTS aux_mercado_sinteticos;
create table IF NOT EXISTS aux_mercado_sinteticos (id int,dif float);
insert into aux_mercado_sinteticos (id,dif) select row_number() over(),close-
open from mercado_sinteticos;

drop table IF EXISTS aux_mercado_sinteticos1;
create table IF NOT EXISTS aux_mercado_sinteticos1 (id int,dif float);
drop table IF EXISTS aux_mercado_sinteticos2;
create table IF NOT EXISTS aux_mercado_sinteticos2 (id int,predic float);

insert into aux_mercado_sinteticos1 (id,dif) select row_number() over(),dif
from aux_mercado_sinteticos where aux_mercado_sinteticos.id >=1   and
aux_mercado_sinteticos.id <='${hiveconf:longitud}'-1;

insert into aux_mercado_sinteticos2 (id,predic) select row_number()
over(),dif from aux_mercado_sinteticos where aux_mercado_sinteticos.id >=2
and  aux_mercado_sinteticos.id <='${hiveconf:longitud}';

drop table IF EXISTS mercado_sinteticos_calculo;
create table IF NOT EXISTS mercado_sinteticos_calculo AS
select n1.dif,n2.predic
from aux_mercado_sinteticos1 n1
inner join aux_mercado_sinteticos2 n2
on n1.id=n2.id;
```

```
drop table IF EXISTS mercado_sinteticos_calculo;
create table IF NOT EXISTS mercado_sinteticos_calculo AS
select n1.dif,n2.predic
from aux_mercado_sinteticos1 n1
inner join aux_mercado_sinteticos2 n2
on n1.id=n2.id;

drop table IF EXISTS aux_mercado_sinteticos_calculo;
create table IF NOT EXISTS aux_mercado_sinteticos_calculo AS
select dif*predic as aux_resultado
from mercado_sinteticos_calculo;

drop table IF EXISTS resultado_mercado_sinteticos;
create table IF NOT EXISTS resultado_mercado_sinteticos (resultado float);
insert into resultado_mercado_sinteticos (resultado) select count(*) from
aux_mercado_sinteticos_calculo where aux_resultado>=0;

select (resultado/('${hiveconf:longitud}'-1))*100 from
resultado_mercado_sinteticos;
```

2.4.3. Pseudo-distributed configuration

On this section, we are going to study the efficiency of the Hadoop infrastructure applying the pseudo-distributed configuration, where we only have one computer that acts as master and slave.

2.4.3.1. Analysis with data from the financial markets

Processing time of the basic algorithm

To calculate the processing time required to run the script that has been programmed, the following command is used:

time hive -f algoritmo_base_(nombre del mercado que queremos medir).hql

The processing times that have been calculated for the different Stock Exchanges are the following:

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	4m50.258s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	4m50.748s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	4m47.065s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	4m44.703s

From the table above, we can conclude that the amount of data only impacts slightly the processing time as we are using small populations.

As shown later on our study, the start of the system Hadoop as well as the Mapreduce processes are the main factors affecting the processing time.

Processing time of the advanced algorithm

To calculate the processing time required to run the script that has been programmed, the following command is used:

time hive -f algoritmo_avanzado.hql

In this case, we only obtain one processing time, as the algorithm itself combines all the different markets object of our analysis and summarizes all the data into one file in Hadoop.

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_avanzado.hql	1845,2 kB	Nasdaq Nikkei Ibex 35	23252	5m3.531s

The processing time when applying the advanced algorithm is higher than when applying the basic algorithm due mainly to the following reasons:

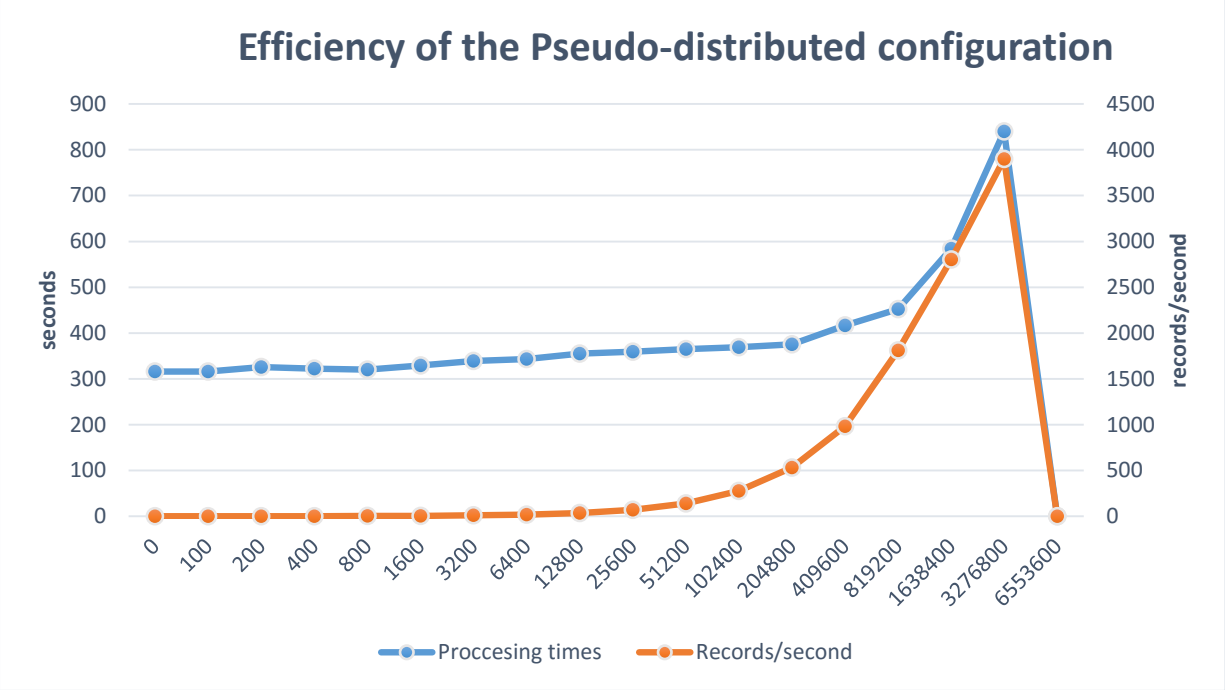
- The data population is bigger in this case (23252 vs 11388 on the biggest population on the basic algorithm)

- The size of the file is bigger as well as it condenses the data from different charts (corresponding to the different markets) into one file, which requires as well more reading operations.

- A higher computational effort is required by the system to create and merge tables required to summarize and combine the data obtained from the different markets.

2.4.3.2. Analysis with synthetic market data

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	5m15.879s	0
100	Nasdaq_1.csv	5m16.275s	0,31
200	Nasdaq_2.csv	5m25.790s	0,61
400	Nasdaq_3.csv	5m22.563s	1,24
800	Nasdaq_4.csv	5m20.313s	2,49
1600	Nasdaq_5.csv	5m29.341s	4,85
3200	Nasdaq_6.csv	5m39.005s	9,44
6400	Nasdaq_7.csv	5m43.531s	18,63
12800	Nasdaq_8.csv	5m55.236s	36,03
25600	Nasdaq_9.csv	5m59.000s	71,31
51200	Nasdaq_10.csv	6m4.942s	140,30
102400	Nasdaq_11.csv	6m9.371s	277,22
204800	Nasdaq_12.csv	6m25.630s	531,08
409600	Nasdaq_13.csv	6m56.382s	983,71
819200	Nasdaq_14.csv	7m32.533s	1810,25
1638400	Nasdaq_15.csv	9m44.362s	2803,74
3276800	Nasdaq_16.csv	14m0.176s	3900,13
6553600	Nasdaq_17.csv	Execution not finalized	0

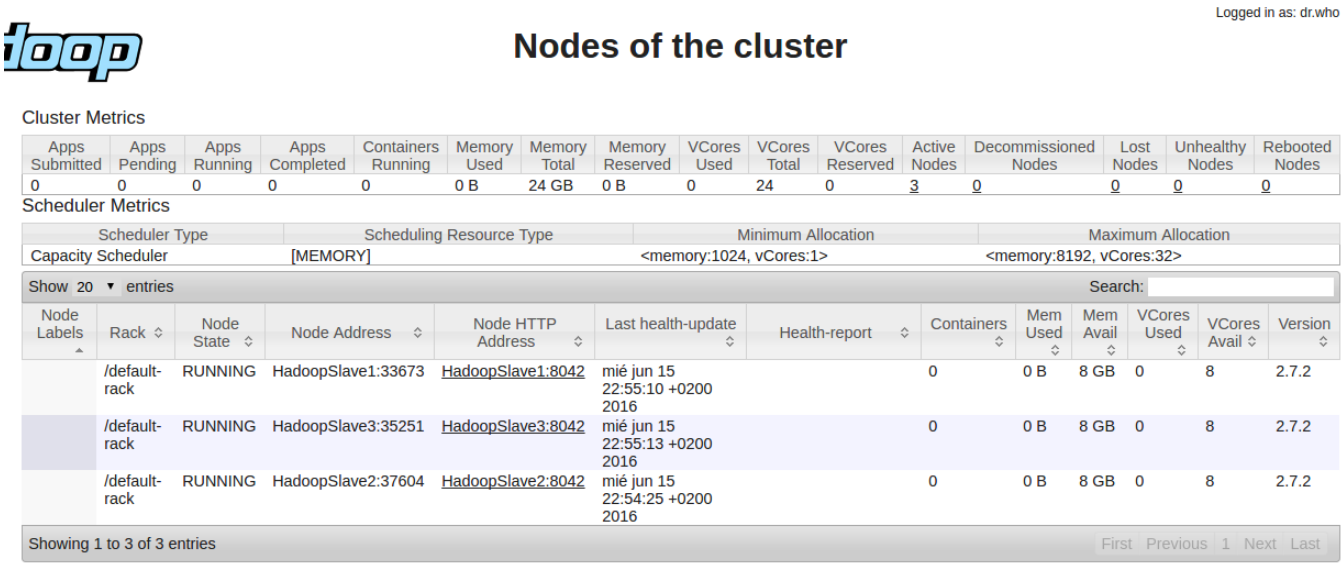


From the graph above, we can see how the processing time remains stable up to 51,200 line items. From this point, the processing time raises sharply until 6,553,600 line items, point where the process couldn't finalize the execution due to lack of RAM memory in the system.

The number of records registered per second follows the same trend as the processing time and the bigger the amount of data received from the system, the higher the efficiency obtained, up to a point where the process cannot be finalized.

2.4.4. Domestic cluster

On this section we are going to follow the same process as in the one above but in this case applying the multinode configuration, where one master and "n" slaves exist depending on the number of nodes being used, all of them in different computers. Since the system is on we can see actives nodes:



Logged in as: dr.who

Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	24 GB	0 B	0	24	0	3	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:32>

Show 20 entries Search:

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
	/default-rack	RUNNING	HadoopSlave1:33673	HadoopSlave1:8042	mié jun 15 22:55:10 +0200 2016		0	0 B	8 GB	0	8	2.7.2
	/default-rack	RUNNING	HadoopSlave3:35251	HadoopSlave3:8042	mié jun 15 22:55:13 +0200 2016		0	0 B	8 GB	0	8	2.7.2
	/default-rack	RUNNING	HadoopSlave2:37604	HadoopSlave2:8042	mié jun 15 22:54:25 +0200 2016		0	0 B	8 GB	0	8	2.7.2

Showing 1 to 3 of 3 entries First Previous 1 Next Last

2.4.4.1. Analysis with data from the financial markets

The processing times obtained for the different markets are the ones shown below:

2.4.4.1.1. One node

Processing time when applying a basic algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m32.548s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	1m2.692s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	0m57.472s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	0m55.434s

Processing time when applying an advanced algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	1m57.353s

2.4.4.1.2. Two nodes

Processing time when applying a basic algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m4.153s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	1m0.523s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	1m0.606s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	1m2.584s

Processing time when applying an advanced algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	2m2.940s

2.4.4.1.3. Three nodes

Processing time when applying a basic algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_base_simplificado.hql	840,7 kB	Nasdaq	11388	1m4.641s
algoritmo_base_simplificado.hql	625,5 kB	Ftse	8408	0m57.628s
algoritmo_base_simplificado.hql	565 kB	Nikkei	7949	0m58.165s
algoritmo_base_simplificado.hql	439,5 kB	Ibex 35	5813	0m56.843s

Processing time when applying an advanced algorithm

Algorithm	Size of the file	Stock Exchange	Data population	Processing time
algoritmo_avanzado.hql	1845,2 kB	Ibex 35 Nasdaq Nikkei	23252	1m50.098s

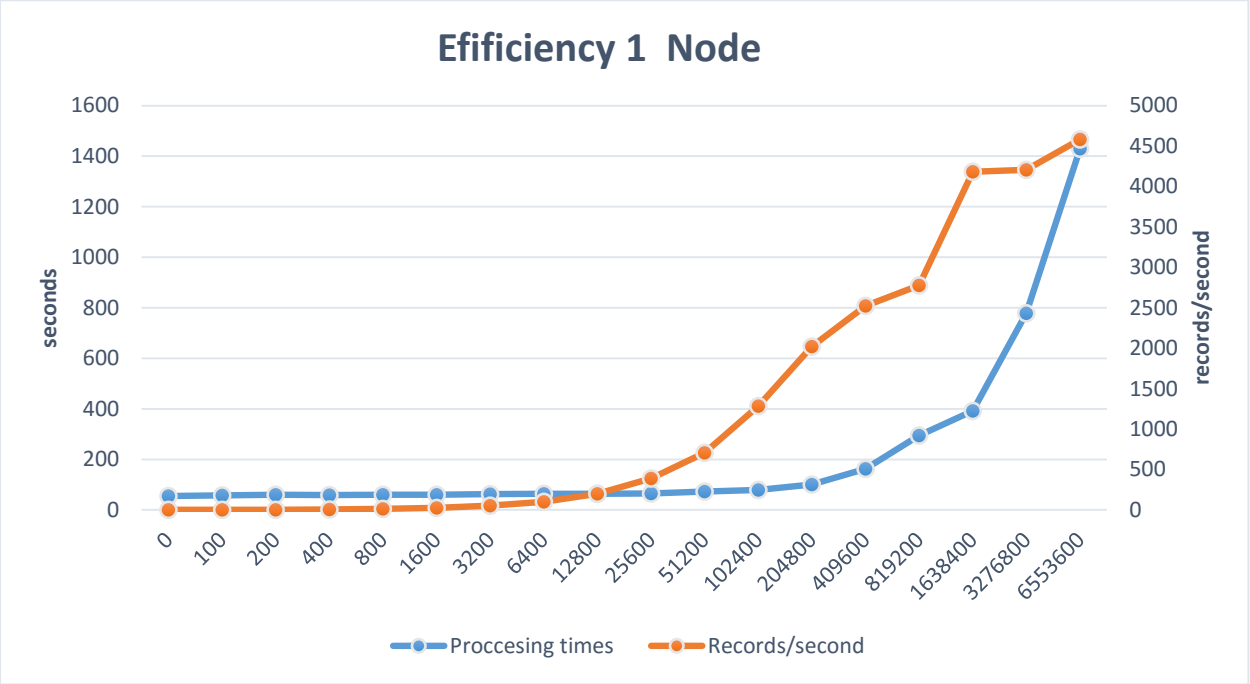
Just as in the pseudo-distributed configuration, the processing time is not affected by the data population as we are using a small amount of data.

It should be pointed out that the performance obtained for two nodes is worse than the one for one and three nodes. We will analyse the causes of this behaviour when we finalize and conclude the full analysis.

2.4.4.2. Analysis with synthetic market data

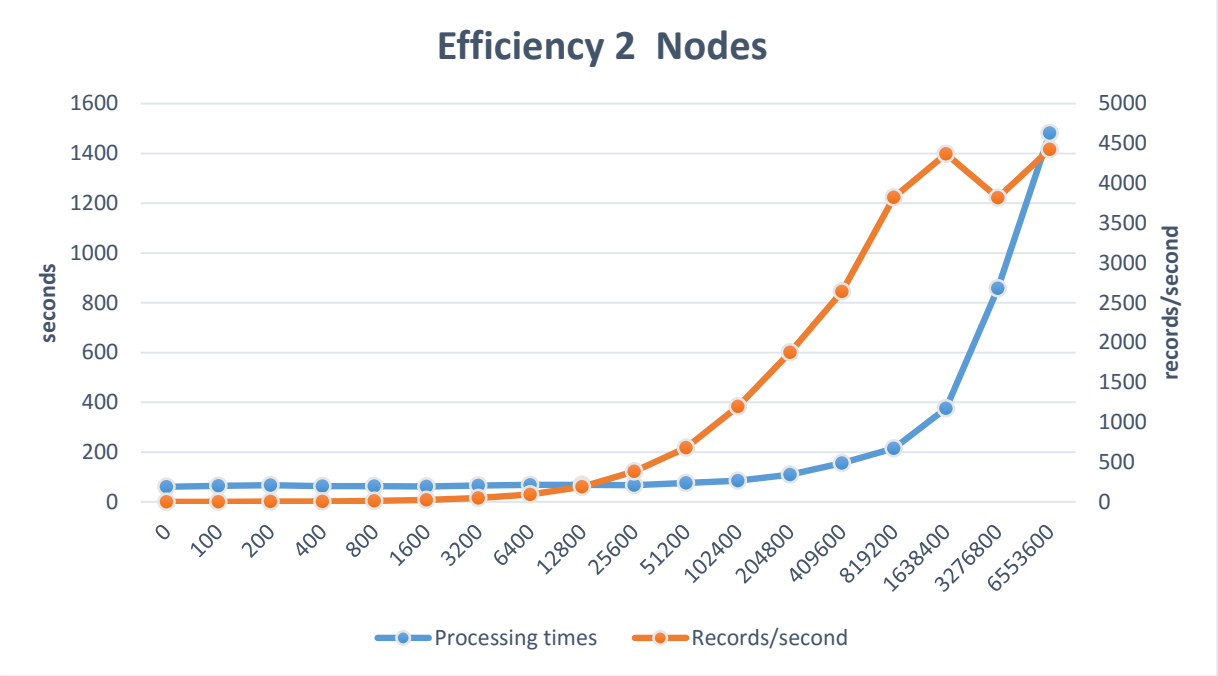
2.4.4.2.1. One node

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m55.629s	0
100	Nasdaq_1.csv	0m57.993s	1,72
200	Nasdaq_2.csv	0m59.707s	3,35
400	Nasdaq_3.csv	0m59.383s	6,74
800	Nasdaq_4.csv	0m59.737s	13,4
1600	Nasdaq_5.csv	1m0.358s	26,65
3200	Nasdaq_6.csv	1m2.223s	51,42
6400	Nasdaq_7.csv	1m3.519s	100,75
12800	Nasdaq_8.csv	1m4.369s	198,85
25600	Nasdaq_9.csv	1m5.404s	391,41
51200	Nasdaq_10.csv	1m12.410s	707,02
102400	Nasdaq_11.csv	1m19.677s	1285,18
204800	Nasdaq_12.csv	1m41.244s	2022,83
409600	Nasdaq_13.csv	2m42.269s	2524,203
819200	Nasdaq_14.csv	4m55.067s	2776,32
1638400	Nasdaq_15.csv	6m31.814s	4181,57
3276800	Nasdaq_16.csv	12m58.855s	4207,20
6553600	Nasdaq_17.csv	23m49.937s	4583,14



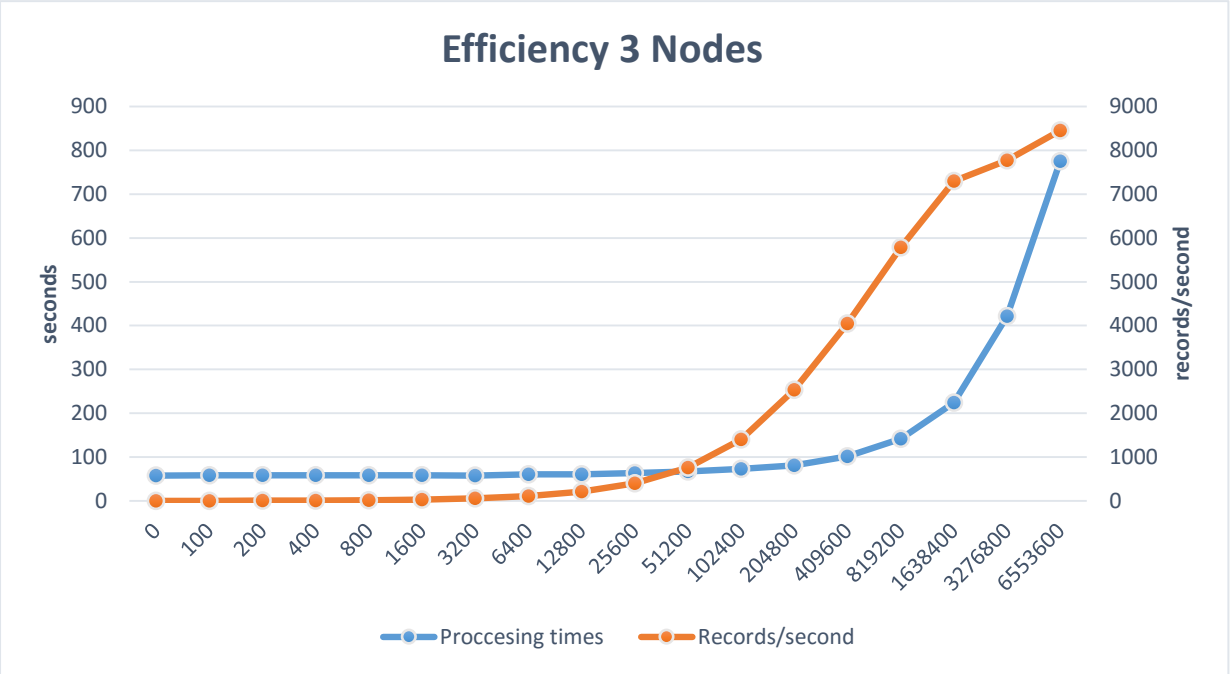
2.4.4.2.2. Two nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	1m0.524s	0
100	Nasdaq_1.csv	1m4.624s	1,54
200	Nasdaq_2.csv	1m7.160s	2,97
400	Nasdaq_3.csv	1m2.778s	6,37
800	Nasdaq_4.csv	1m3.152s	12,66
1600	Nasdaq_5.csv	1m2.143s	25,74
3200	Nasdaq_6.csv	1m6.148s	48,37
6400	Nasdaq_7.csv	1m8.220s	93,81
12800	Nasdaq_8.csv	1m8.232s	187,6
25600	Nasdaq_9.csv	1m7.224s	380,81
51200	Nasdaq_10.csv	1m15.465s	678,46
102400	Nasdaq_11.csv	1m25.563s	1196,78
204800	Nasdaq_12.csv	1m48.988s	1879,105
409600	Nasdaq_13.csv	2m35.186s	2639,41
819200	Nasdaq_14.csv	3m34.326s	3822,21
1638400	Nasdaq_15.csv	6m15.112s	4367,76
3276800	Nasdaq_16.csv	14m18.578s	3816,54
6553600	Nasdaq_17.csv	24m41.552s	4423,47



2.4.4.2.3. Three nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m57.430s	0
100	Nasdaq_1.csv	0m58.172s	1,72
200	Nasdaq_2.csv	0m58.703s	3,40
400	Nasdaq_3.csv	0m58.339s	6,85
800	Nasdaq_4.csv	0m58.410s	13,7
1600	Nasdaq_5.csv	0m58.069s	27,55
3200	Nasdaq_6.csv	0m57.653s	55,5
6400	Nasdaq_7.csv	1m0.234s	106,62
12800	Nasdaq_8.csv	1m0.617s	213,12
25600	Nasdaq_9.csv	1m3.764s	401,48
51200	Nasdaq_10.csv	1m7.333s	760,4
102400	Nasdaq_11.csv	1m12.957s	1403,56
204800	Nasdaq_12.csv	1m20.777s	2535,37
409600	Nasdaq_13.csv	1m41.254s	4045,27
819200	Nasdaq_14.csv	2m21.537s	5787,88
1638400	Nasdaq_15.csv	3m44.568s	7295,78
3276800	Nasdaq_16.csv	7m1.354s	7776,83
6553600	Nasdaq_17.csv	12m55.394s	8451,96



2.4.5. Universitary cluster

We are going to do the same method tan in the domestic cluster, but this time for a multinode configuration based on a universitary cluster with one master and whit 24 slaves maximum, all of them in different computers. Since the system is on we can see actives nodes:

Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0 B	192 GB	0 B	0	192	0	24	0	0	0	0

trics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
duler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

ntries Search:

Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack	RUNNING	it003.lab.it.uc3m.es:56213	it003.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	doc026.lab.it.uc3m.es:58702	doc026.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it014.lab.it.uc3m.es:52865	it014.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm006.lab.it.uc3m.es:45624	lm006.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it010.lab.it.uc3m.es:50011	it010.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it007.lab.it.uc3m.es:52930	it007.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it011.lab.it.uc3m.es:46619	it011.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it020.lab.it.uc3m.es:32769	it020.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it017.lab.it.uc3m.es:60462	it017.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm008.lab.it.uc3m.es:44031	lm008.lab.it.uc3m.es:8042	vie jun 17 19:39:47 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it016.lab.it.uc3m.es:55966	it016.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm010.lab.it.uc3m.es:45439	lm010.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it002.lab.it.uc3m.es:51516	it002.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it012.lab.it.uc3m.es:60649	it012.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it005.lab.it.uc3m.es:47216	it005.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it019.lab.it.uc3m.es:53957	it019.lab.it.uc3m.es:8042	vie jun 17 19:39:55 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm001.lab.it.uc3m.es:44105	lm001.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it009.lab.it.uc3m.es:56852	it009.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it008.lab.it.uc3m.es:49346	it008.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it013.lab.it.uc3m.es:49490	it013.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	lm009.lab.it.uc3m.es:46873	lm009.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it015.lab.it.uc3m.es:59819	it015.lab.it.uc3m.es:8042	vie jun 17 19:39:46 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it018.lab.it.uc3m.es:42131	it018.lab.it.uc3m.es:8042	vie jun 17 19:39:49 +0200 2016		0	0 B	8 GB	0	8	2.7.2
/default-rack	RUNNING	it006.lab.it.uc3m.es:53930	it006.lab.it.uc3m.es:8042	vie jun 17 19:39:48 +0200 2016		0	0 B	8 GB	0	8	2.7.2

t of 24 entries First Previous 1 Next Last

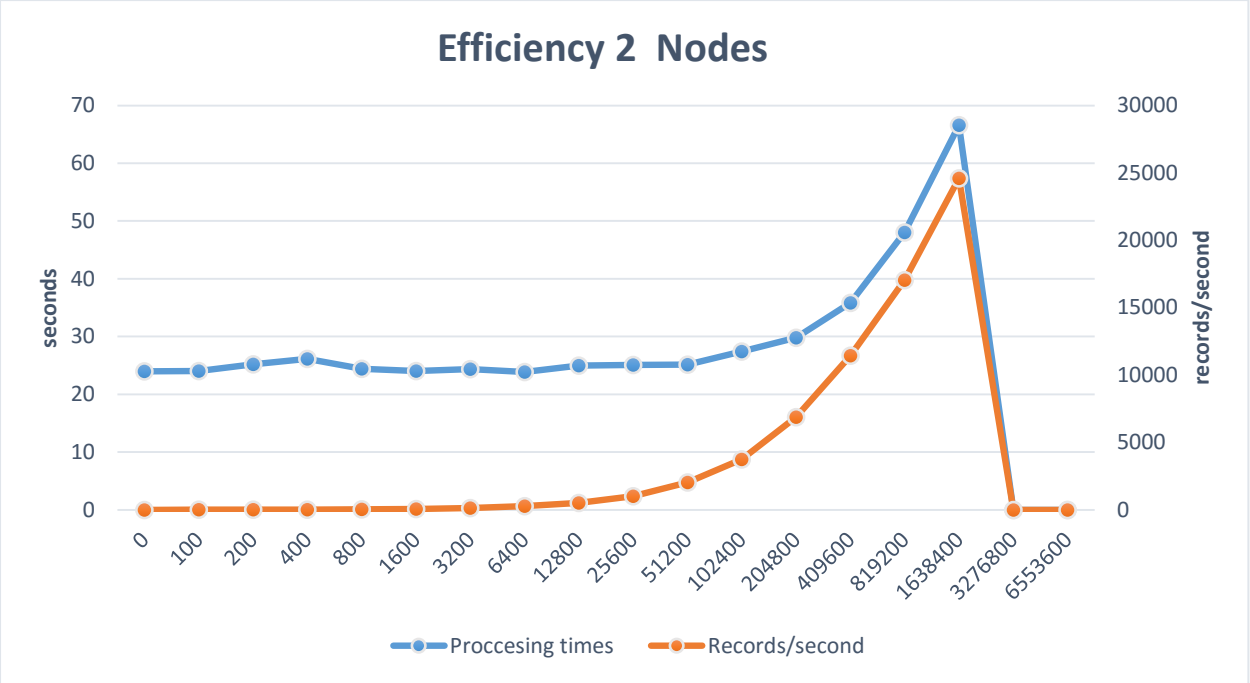
We do this configuration with scripts and doing ssh connections to all of the computers of the cluster (attached II)

At this time, we are going to analyze the efficiency of the system only with the synthetic data.

The limitation of the universitary cluster is the amount of storage that we can use on each computer. The storage provided by the Telematics department are 2,8 GB for the strict case, and 3,2 GB for the extended case. For this reason, we cannot process Nasdaq_16.csv and Nasdaq_17.csv files

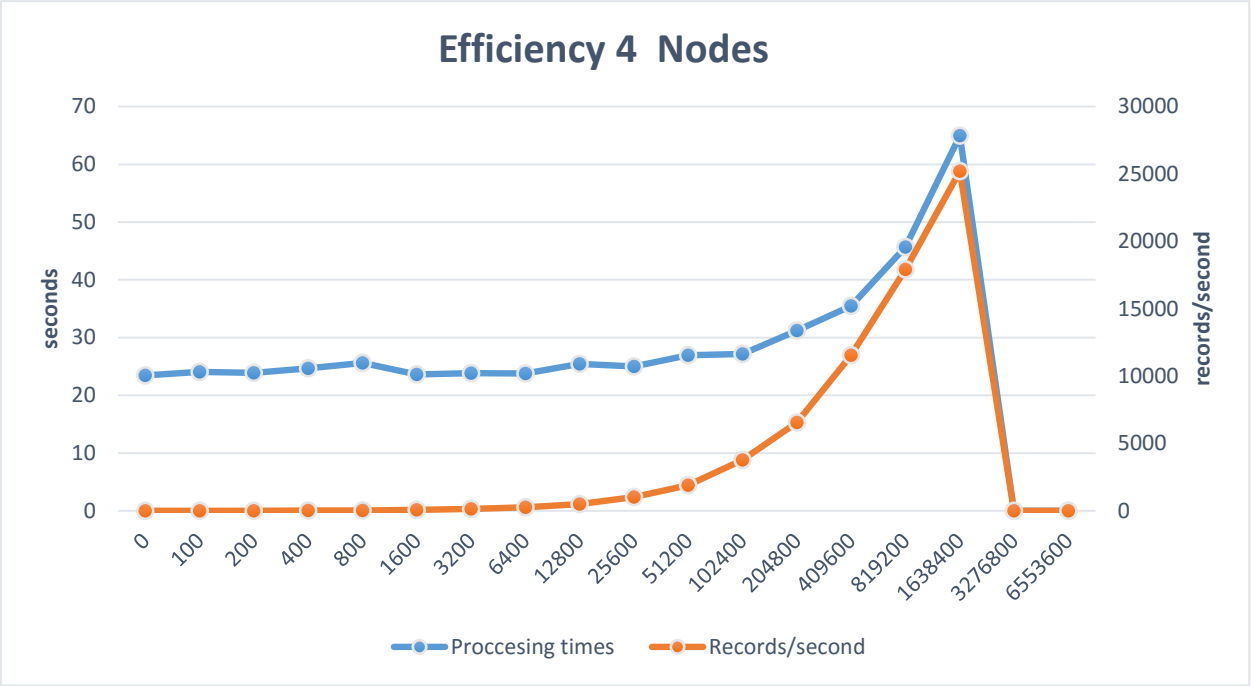
2.4.5.1. 2 slave nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m23.987s	0
100	Nasdaq_1.csv	0m24.024s	4,16
200	Nasdaq_2.csv	0m25.221s	7,93
400	Nasdaq_3.csv	0m26.128s	15,31
800	Nasdaq_4.csv	0m24.407s	32,77
1600	Nasdaq_5.csv	0m24.035s	66,57
3200	Nasdaq_6.csv	0m24.383s	131,24
6400	Nasdaq_7.csv	0m23.843s	268,42
12800	Nasdaq_8.csv	0m24.946s	513,11
25600	Nasdaq_9.csv	0m25.109s	1019,55
51200	Nasdaq_10.csv	0m25.143s	2036,35
102400	Nasdaq_11.csv	0m27.417s	3734,91
204800	Nasdaq_12.csv	0m29.807s	6870,87
409600	Nasdaq_13.csv	0m35.837s	11429,52
819200	Nasdaq_14.csv	0m48.035s	17054,23
1638400	Nasdaq_15.csv	1m6.586s	24605,77
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0



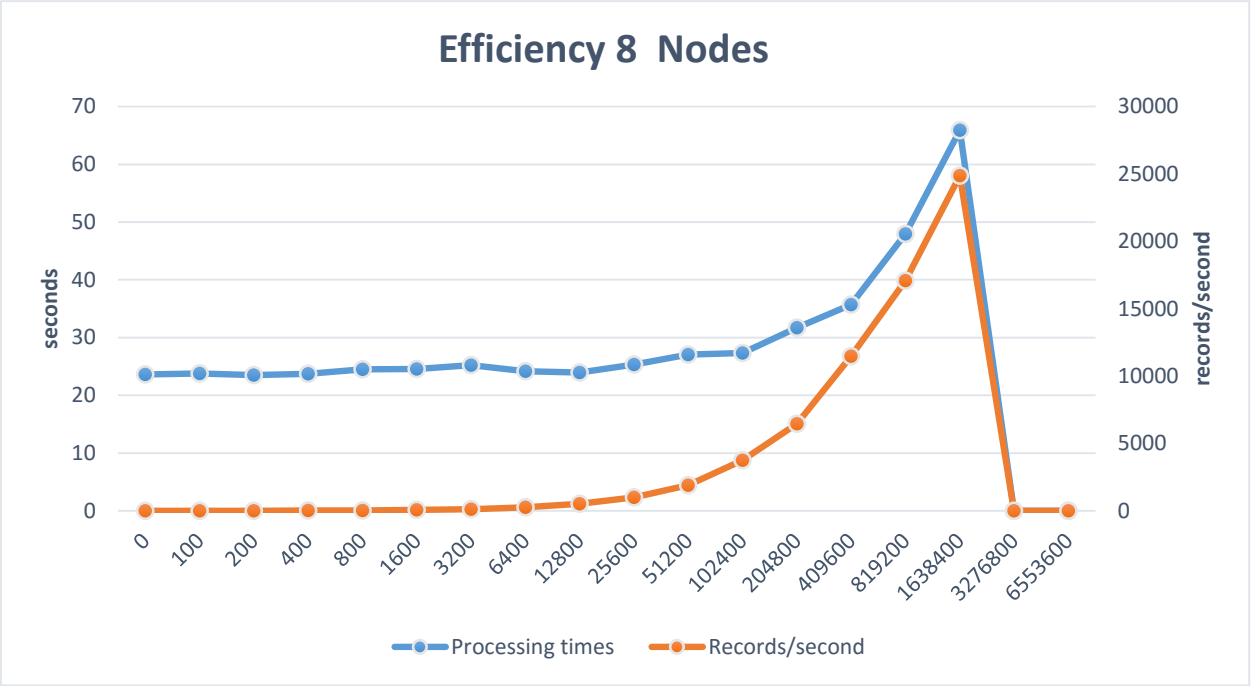
2.4.5.2. 4 slave nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m23.469s	0
100	Nasdaq_1.csv	0m24.075s	4,15
200	Nasdaq_2.csv	0m23.921s	8,36
400	Nasdaq_3.csv	0m24.694s	16,2
800	Nasdaq_4.csv	0m25.598s	31,25
1600	Nasdaq_5.csv	0m23.604s	67,78
3200	Nasdaq_6.csv	0m23.822s	134
6400	Nasdaq_7.csv	0m23.773s	269,22
12800	Nasdaq_8.csv	0m25.440s	503,14
25600	Nasdaq_9.csv	0m25.015s	1023,38
51200	Nasdaq_10.csv	0m26.963s	1898,89
102400	Nasdaq_11.csv	0m27.157s	3770,66
204800	Nasdaq_12.csv	0m31.207s	6562,63
409600	Nasdaq_13.csv	0m35.500s	11538,02
819200	Nasdaq_14.csv	0m45.706s	17923,24
1638400	Nasdaq_15.csv	1m4.974s	25216,24
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0



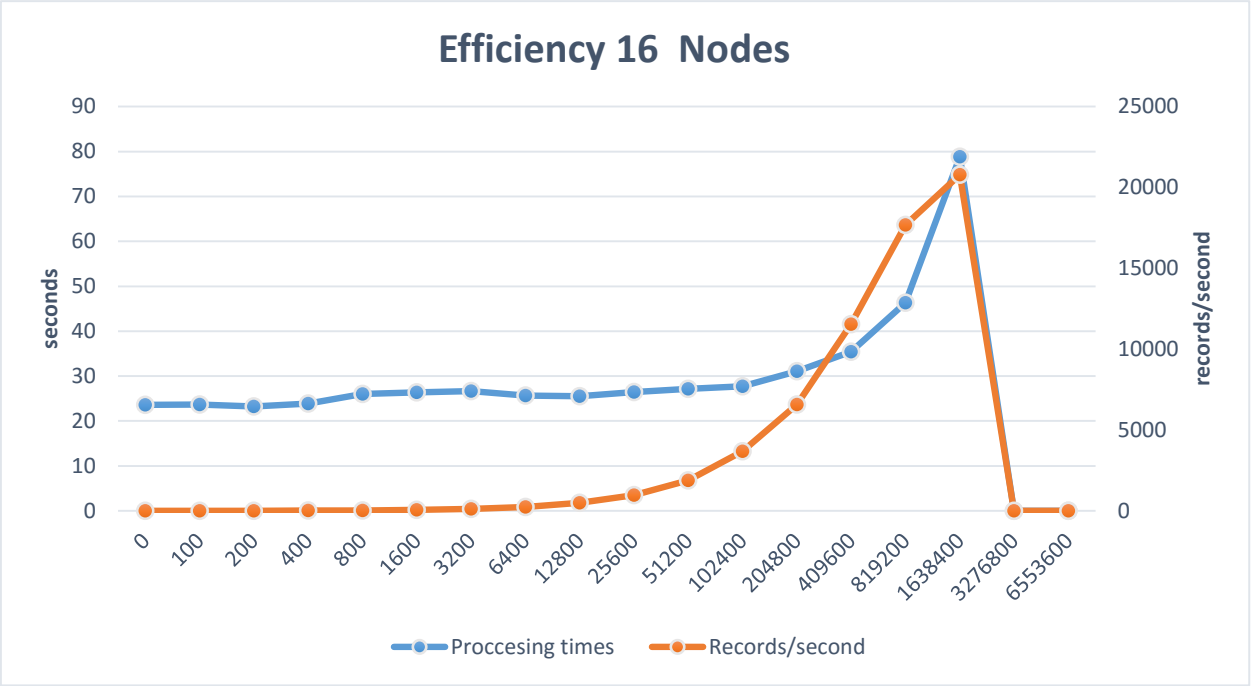
2.4.5.3. 8 slave nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m23.614s	0
100	Nasdaq_1.csv	0m23.785s	4,20
200	Nasdaq_2.csv	0m23.508s	8,50
400	Nasdaq_3.csv	0m23.737s	16,85
800	Nasdaq_4.csv	0m24.529s	32,61
1600	Nasdaq_5.csv	0m24.553s	65,16
3200	Nasdaq_6.csv	0m25.226s	126,85
6400	Nasdaq_7.csv	0m24.195s	264,51
12800	Nasdaq_8.csv	0m23.939s	534,69
25600	Nasdaq_9.csv	0m25.362s	1009,38
51200	Nasdaq_10.csv	0m27.072s	1891,25
102400	Nasdaq_11.csv	0m27.334s	3746,25
204800	Nasdaq_12.csv	0m31.704s	6459,75
409600	Nasdaq_13.csv	0m35.697s	11474,35
819200	Nasdaq_14.csv	0m47.957s	17081,97
1638400	Nasdaq_15.csv	1m5.891s	24865,30
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0



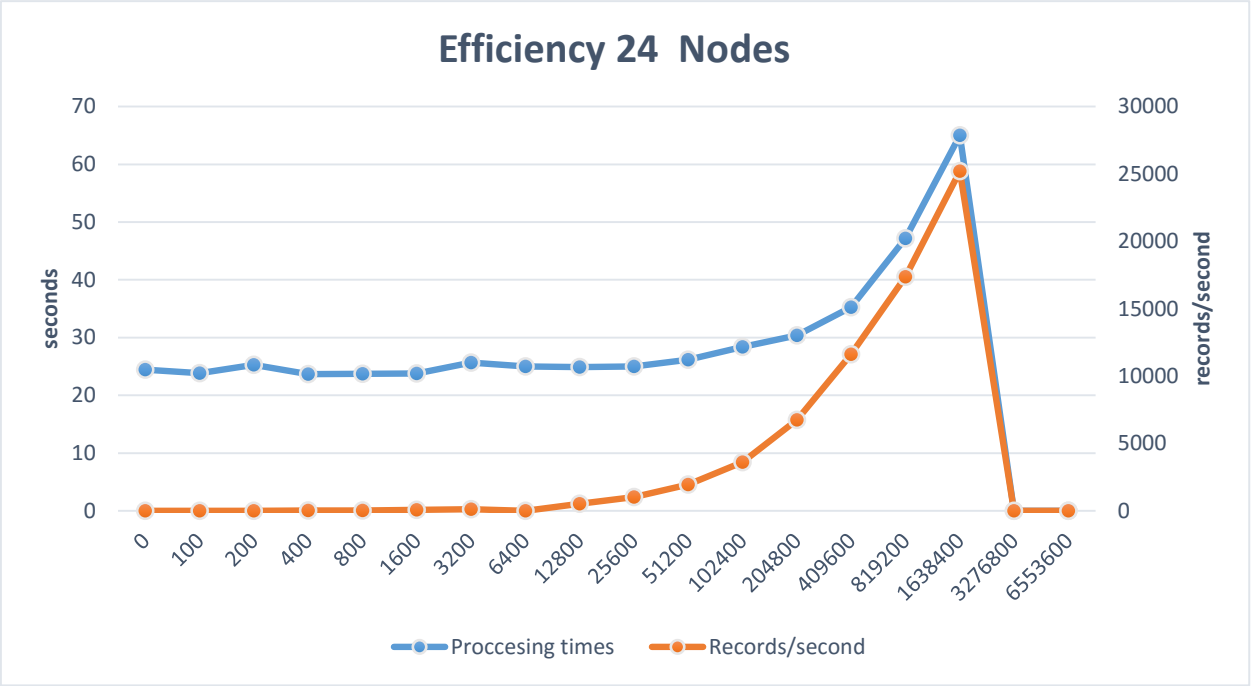
2.4.5.4. 16 slave nodes

Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m23.592s	0
100	Nasdaq_1.csv	0m23.652s	4,22
200	Nasdaq_2.csv	0m23.225s	8,61
400	Nasdaq_3.csv	0m23.890s	16,74
800	Nasdaq_4.csv	0m26.049s	30,71
1600	Nasdaq_5.csv	0m26.410s	60,58
3200	Nasdaq_6.csv	0m26.651s	120,07
6400	Nasdaq_7.csv	0m25.696s	249,066
12800	Nasdaq_8.csv	0m25.541s	501,15
25600	Nasdaq_9.csv	0m26.438s	968,30
51200	Nasdaq_10.csv	0m27.157s	1885,33
102400	Nasdaq_11.csv	0m27.762s	3688,49
204800	Nasdaq_12.csv	0m31.106s	6583,93
409600	Nasdaq_13.csv	0m35.453s	11553,32
819200	Nasdaq_14.csv	0m46.345s	17676,12
1638400	Nasdaq_15.csv	1m18.814s	20788,18
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0



2.4.5.5. 24 slave nodes

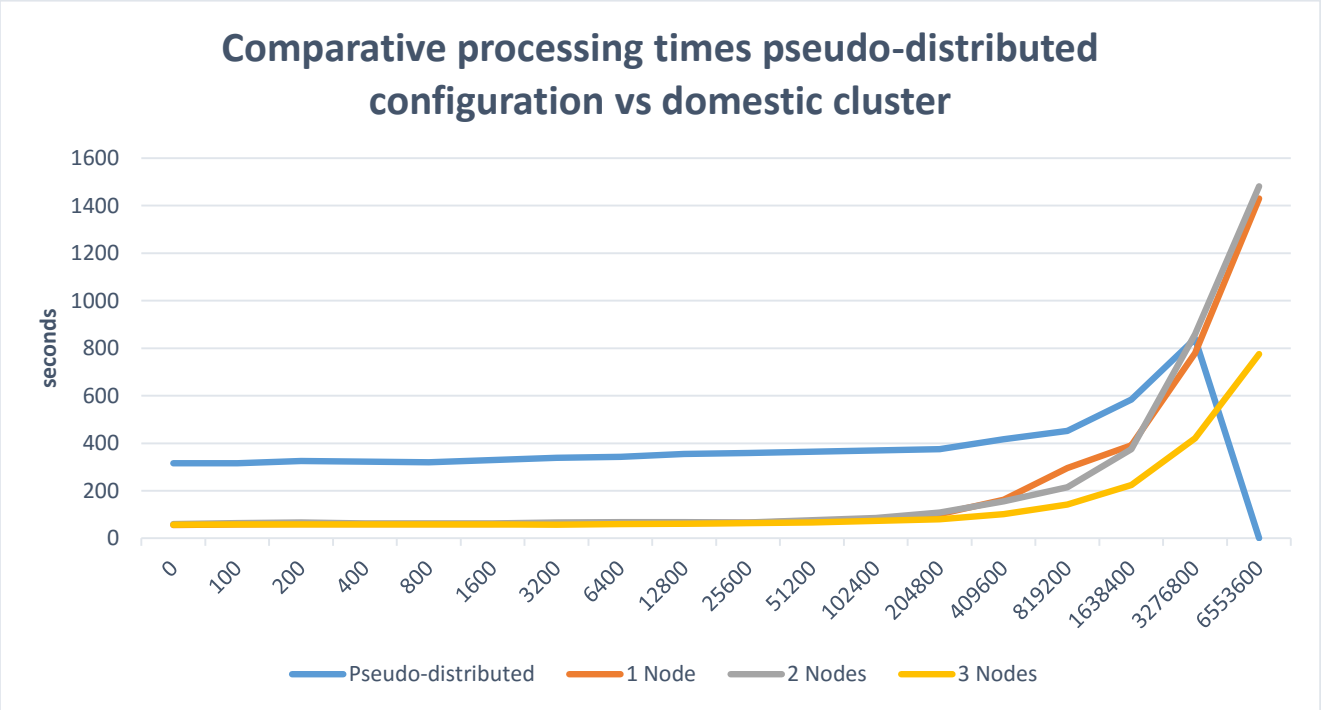
Data population (line items)	Name of the data file	Processing times	Records/Second
0	Nasdaq_0.csv	0m24.449s	0
100	Nasdaq_1.csv	0m23.831s	4,19
200	Nasdaq_2.csv	0m25.298s	7,90
400	Nasdaq_3.csv	0m23.689s	16,88
800	Nasdaq_4.csv	0m23.717s	33,73
1600	Nasdaq_5.csv	0m23.801s	67,22
3200	Nasdaq_6.csv	0m25.643s	124,79
6400	Nasdaq_7.csv	0m25.014s	255.85
12800	Nasdaq_8.csv	0m24.866s	514,76
25600	Nasdaq_9.csv	0m25.013s	1023,47
51200	Nasdaq_10.csv	0m26.161s	1957,11
102400	Nasdaq_11.csv	0m28.369s	3609,57
204800	Nasdaq_12.csv	0m30.363s	6745,05
409600	Nasdaq_13.csv	0m35.256s	11617,88
819200	Nasdaq_14.csv	0m47.186s	17361,08
1638400	Nasdaq_15.csv	1m5.012s	25201,50
3276800	Nasdaq_16.csv	0	0
6553600	Nasdaq_17.csv	0	0



2.4.6. Comparison between the efficiencies obtained on the different configurations on the Hadoop architecture

In order to compare the different efficiencies obtained on the different configurations used, we have summarized, as shown on the graph below, the results from the previous sections in one diagram.

2.4.6.1. Processing times



As seen in the graph above, the configuration that shows the highest efficiency in terms of processing time is the one composed of 3 slave nodes and 1 master, while the one showing the least efficiency of all is the pseudo-distributed configuration.

The main conclusion we obtain from our analysis is that a minimum amount of data is required (204,800 records) to reach efficiency in terms of processing time so that the amount of data being processed is higher in a smaller amount of time (exponential growth). Also, the benefits of having a multi-node architecture with a bigger number of nodes are only seen when processing a minimum amount of records.

In general, we can conclude that the multinode architecture is more efficient in terms of processing time than the pseudo-distributed configuration, independently of the number of records processed. The reason is that the processing power is higher on the first one as the number of processors is bigger than on a pseudo-distributed configuration.

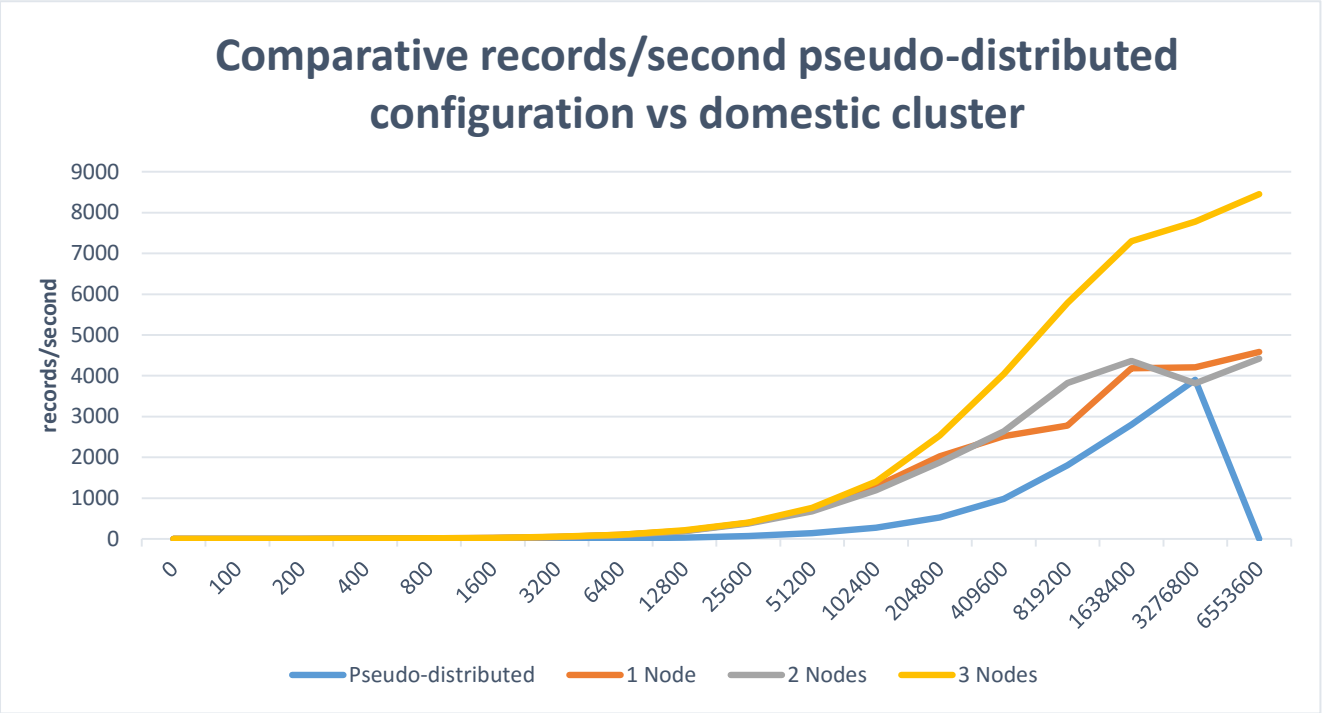
Also, it should be highlighted that the amount of records that can be processed in a multinode configuration is bigger the higher the amount of nodes used on the configuration, maintaining the same clock rate of the processors. This means that the amount of records that can be processed by two nodes would be bigger than the ones that could be processed when using a single node as the capacity of processing increases but the processing time and speed remain stable.

However, the four configurations have the following points in common:

- Up to 204,800 records, the processing time is very similar, despite the increase on the amount of data. We don’t see an increase on the processing time until we double the data population until 409,600 records. Therefore, the efficiency starts to be interesting only when we reach the point on which the processing time starts to increase significantly as up to this point the temporary cost of processing the data is stable.

- Once we have reached the minimum level of 204,800 records, we see that the processing time starts to increase proportionally to the amount of records processed.

2.4.6.2. Records per second



The graph above shows the efficiency of the architecture in terms of records processed per second when varying the amount of data processed. The number of records processed remains stable up to 25,600 line items. From there on, the amount of records processed per second increases in parallel to the amount of data on the population as well as the processing time.

When running the pseudo-distributed configuration with a data load of **3,276,800** or more, we found that the execution of the additional work load couldn't be finalized as there was no enough RAM memory in the system. This makes this configuration instable and not advisable when considering big populations of data and massive data processing is required.

On the other hand, on the multinode configuration we observe a reduction on the performance of the computer both in the case of one node and two nodes when going from 1,638,400 to 3,276,800 records. This, however, doesn't happen when a third node is added to the configuration. The main reason is the "Communications" factor, as explained below.

We have summarized below the two main conclusions of our study:

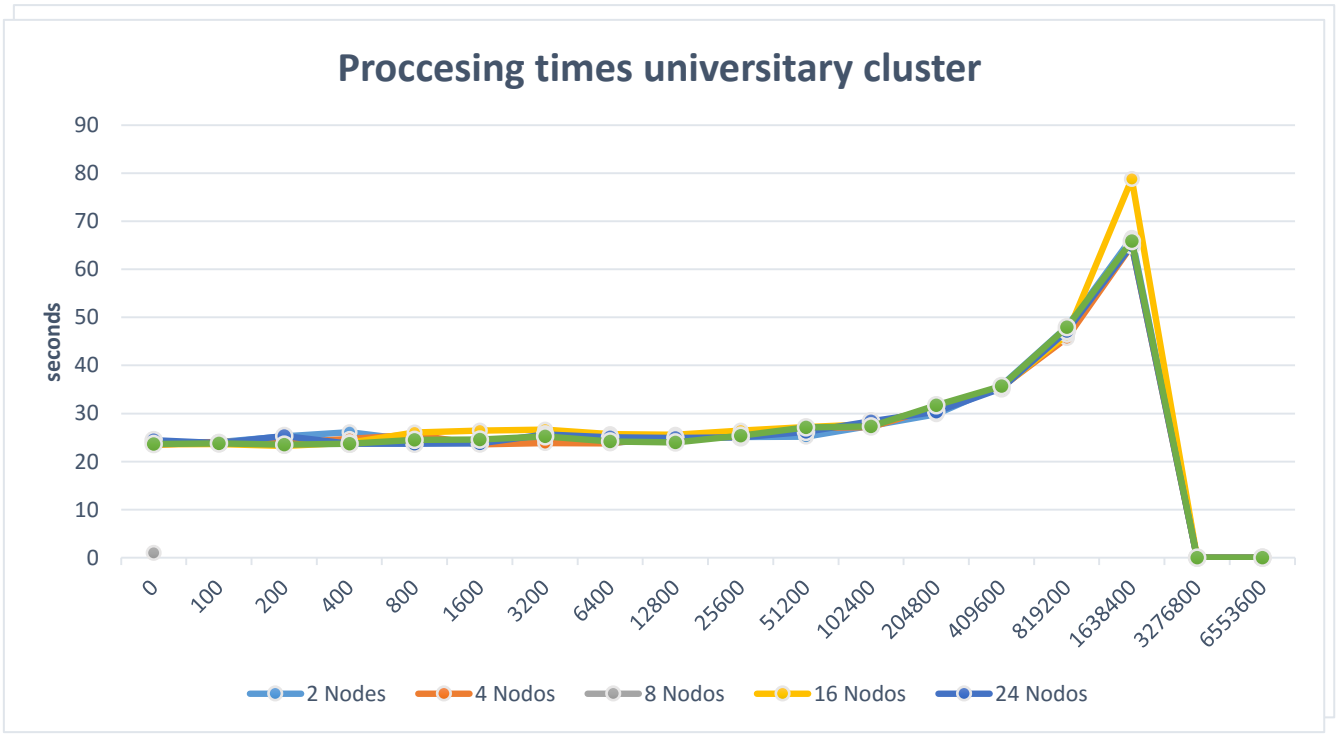
1. The RAM memory is critical: as mentioned above, a limited RAM memory on a pseudo-distributed configuration prevents the processing of a big amount of records, making the system useless when considering the analysis of big financial markets with an extensive amount of data.

2. Number of nodes vs processing time, records per second: looking at the behaviour of our system with one and two nodes, it is important to highlight the worsening on the performance of the system when two nodes are being used, even though we count with one more computer and therefore a higher computational power thanks to the processor and RAM memory of the new node.

This is explained by the "Communications factor", meaning that we are being limited by the network in comparison with the performance added by the new computer; the delivery of data from the master to the slaves has a higher cost than the processing capacity of the new computer and therefore a reduction on the performance is observed, as shown on the graph above.

Therefore, when designing a Hadoop architecture, it is critical to consider the factors explained above. In general, the number of computers is as important as the number of records to be processed as well as the performance of the nodes available on the system and the "communications" factor.

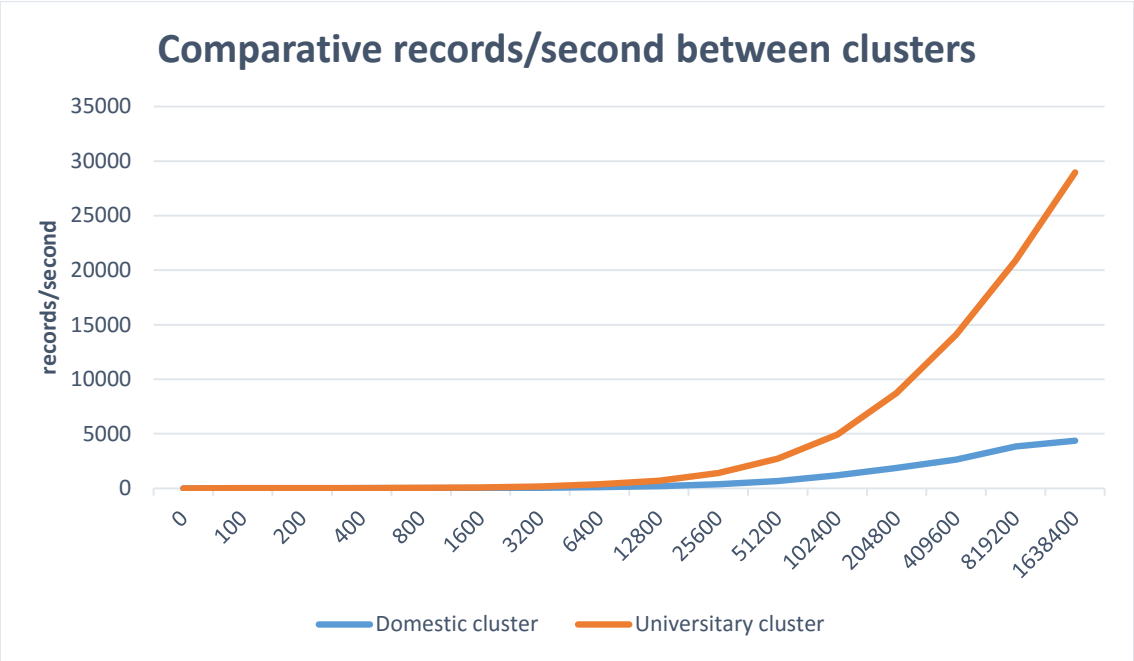
With the intention of corroborating the conclusions obtained as a result of our project, we are going now to test our infrastructure on a university environment (cluster), where the number of computers available is higher than on the initial scenario considered as part of our study (domestic cluster). We will analyse the efficiency of the architecture when using different algorithms on a multinode configuration.

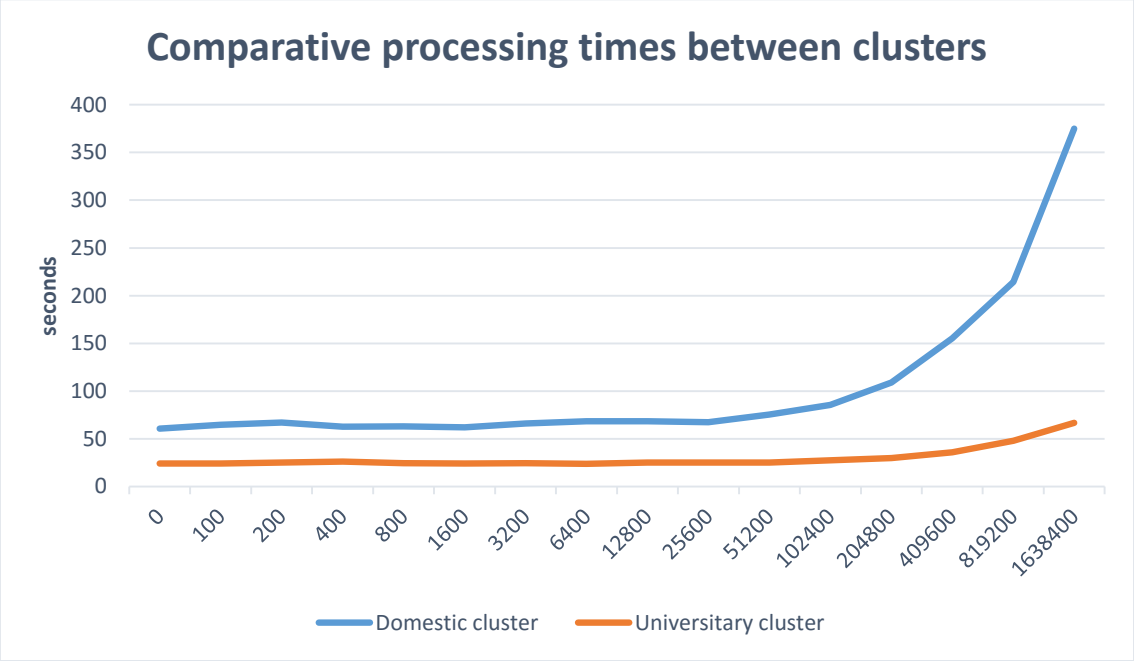


From both charts above, we can endorse the initial conclusions obtained as a result of our project.

On one hand, we see on the tables included on the section 2.4.5 (Attached I) how the output in terms of records per second improves from two to four nodes, worsens from four to eight nodes and reaches a breakeven point on sixteen nodes, where the capacity drops considerably. Once this point has been exceeded and 24 slave nodes are added to our infrastructure, we can observe again a significant improvement on the efficiency of our architecture. This behaviour is mainly explained by the relationship between the number of computers, the RAM memory and the "communications" factor, previously explained on our study.

On the other hand, when comparing the domestic cluster with the university, we discover that the communications factor impacts more severely than expected the performance of the Big Data system that has been generated, as we can see on the graphs on the following page. As shown on these graphs, on the university cluster we obtain shorter and better processing times leading to an improvement on the performance of the system in terms of records per second. Both clusters count with the same amount of slave nodes and RAM memory but differ on the type of connection being used for communication between the different computers – optical fiber versus UTP cat 5e cable.


Comparative records/second between clusters
records/second
35000
30000
25000
20000
15000
10000
5000
0
0
100
200
400
800
1600
3200
6400
12800
25600
51200
102400
204800
409600
819200
1638400
Domestic cluster
Universitary cluster


Comparative processing times between clusters
seconds
400
350
300
250
200
150
100
50
0
0
100
200
400
800
1600
3200
6400
12800
25600
51200
102400
204800
409600
819200
1638400
Domestic cluster
Universitary cluster

3.Conclusions and future work lines

3.1. Conclusions

The objective of the project was to design and implement a Big Data system that would, on one hand, allow the storage, transformation and analysis of an extensive amount of data from the financial markets but, on the other hand, would be efficient and scalable, providing a versatile solution, and also adaptable to different scenarios and budgets.

After implementing all the pre-established tasks and concluding our project, we have achieved all the desired outcomes and reached the initial goals. We have designed and implemented an efficient, scalable and versatile infrastructure for analysis of Big Data, in this case with special focus on the financial markets and stock exchanges. Thanks to the Hadoop and Hive technologies, the programmed infrastructure allows the user to, on a very simple and efficient manner, process very complex data and extract reliable results so that further analysis can be executed.

Furthermore, we have been able to detect and identify the strengths and weaknesses of the system by carrying out various stress tests on diverse scenarios with different combinations of configurations and algorithms. We have detailed the results of our testing on an exhaustive and comprehensive manner on different tables and graphs that give a visual snapshot of the main conclusions of our study by comparing the different algorithms generated in terms of efficiency and effectiveness of the infrastructure. This creates the basics of a potential work line to be considered in future projects for further extension of the analysis on financial markets.

3.2. Future work lines

From the work performed and the results obtained, two main lines can be considered in the future to further extend the scope of our project:

- Economic modelling using algorithms: shortening the frequency of the data to be used on the algorithm to a daily basis so that the model can capture all movements occurred on the stock prices on a more frequent basis and therefore consider the dynamism of the changing world we live in. This would therefore allow the analysis and forecasting of also short-term trends and tendencies on the stock markets.
- Architecture:
    - Increase on the number of nodes
    - Use of a cluster with no limitation on the disk space
    - Development of a cluster with a high number of nodes and inclusive of services such as AWS for cloud computing

Anexo II: Procedimiento de configuración cluster universitario

La configuración del cluster universitario frente a la del cluster doméstico requiere de mayor complejidad debido al número de nodos a configurar. Para ello, en este anexo se detallan los procedimientos seguidos para configurar tanto el sistema Big Data como las pruebas a realizar.

Configuración del sistema

Al igual que en otro tipo de configuraciones, es necesario editar los archivos de Hadoop para implementar la solución deseada. En este caso, se modifican los siguientes archivos:

- Fichero de configuración del sistema Hadoop: core-site.xml

$ sudo gedit core-site.xml

```
<configuration>
<property>
  <name>fs.default.name</name>
  <value>hdfs://163.117.144.136:9000</value>
</property>
</configuration>
```

La dirección HDFS corresponde a la dirección IP de la máquina maestro, en este caso 163.117.144.136 apuntando al puerto 9000.

- Fichero de configuración del sistema HDFS: hdfs-site.xml:

$ sudo gedit hdfs-site.xml

```
<configuration>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
 <property>
      <name>dfs.namenode.name.dir</name>
<value>file:/var/home/lab/alum0/02/84/736/hadoop_tmp/hdfs/n
amenode</value>
 </property>
 <property>
      <name>dfs.datanode.data.dir</name>
<value>file:///var/home/lab/alum0/02/84/736/hadoop_tmp/hdfs
/datanode</value>
 </property>
</configuration>
```

Debido a propia arquitectura del laboratorio de telemática de la Universidad Carlos III y a la cuenta de usuario compartida, se establece el factor de replicación (máquinas dónde se replica la información) a 1, es decir, la información solo se encuentra en un nodo a modo de copia de seguridad.

- Fichero de configuración de Mapreduce: mapred-site.xml

$ sudo gedit mapred-site.xml

```
<configuration>
<property>
       <name>mapreduce.job.tracker</name>
       <value>163.117.144.136:5431</value>
</property>
<property>
       <name>mapred.framework.name</name>
       <value>yarn</value>
</property>
</configuration>
```

- Fichero de configuración de los esclavos: slaves

El fichero de configuración de los esclavos tendrá como máximo 24 direcciones IP, que es el número de máquinas disponible. Para elegir el número de esclavos basta con poner en el fichero el conjunto de direcciones IP deseado.

```
163.117.144.137
163.117.144.135
163.117.144.130
163.117.144.216
163.117.144.217
163.117.144.138
163.117.144.139
163.117.144.211
163.117.144.212
163.117.144.154
163.117.144.202
163.117.144.206
163.117.144.207
163.117.144.218
163.117.144.219
163.117.144.220
163.117.144.208
163.117.144.209
163.117.144.210
163.117.144.213
163.117.144.214
163.117.144.215
163.117.144.203
163.117.144.205
```

Configuración del sistema

Al tratarse de una configuración con un elevado número de nodos, buscamos automatizar el proceso de pruebas para obtener un procedimiento semiautomático, tratando de minimizar posibles errores en los cambios de ficheros.

Para conseguir lo anterior, creamos dos scripts:

- Ejecutar_all.sh

```
bash ejecutar.sh >& ejecutar.log
cat ejecutar.log |grep real
cat ejecutar.log | grep real > tiempos.real.log
```

Ejecutar_all.sh es un script bash que se encarga de ejecutar el archivo ejecutar.sh mostrado más adelante, guardando los resultados que se obtendrían por pantalla en el archivo ejecutar.log.

Una vez hemos obtenido el archivo ejecutar.log, filtramos dicho fichero buscando el término 'real', que hace referencia al tiempo total usado por el script algoritmo simplificado. De esta manera obtenemos un archivo final tiempos.real.log en dónde estarán reflejados cada uno de los tiempos empleados por el script algoritmo_simplificado para los archivos creados con los datos sintéticos.

- Ejecutar.sh

```
time hive/bin/hive -f scripts/algoritmo_simplificado_0.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_1.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_2.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_3.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_4.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_5.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_6.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_7.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_8.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_9.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_10.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_11.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_12.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_13.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_14.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_15.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_16.hql
time hive/bin/hive -f scripts/algoritmo_simplificado_17.hql
```

Ejecutar.sh contiene las instrucciones para obtener el tiempo empleado por cada uno de los ficheros de datos sintéticos.